VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Modelování pohotovosti systému metodou Monte Carlo

Availability modeling by Monte Carlo method

2013 Simona Domesová

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Zadání bakalářské práce

Student: **Simona Domesová**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 1103R031 Výpočetní matematika

Téma: Modelování pohotovosti systému metodou Monte Carlo.
Availability modeling by Monte Carlo method

Zásady pro vypracování:

Monte Carlo je simulační metoda, která slouží jako efektivní nástroj k řešení inženýrských problémů zejména tehdy, pokud analytické metody selhávají. Tato práce umožní zvládnutí této metody pro účely modelování a kvantifikace pohotovosti systémů s nezávislými prvky, vybraných z inženýrské praxe.
Postup práce:

1. Studium základů simulačních metod. Princip metody Monte Carlo.
2. Kvantifikace pohotovosti systému s nezávislými prvky analyticky.
3. Řešení pomocí metody Monte Carlo. Srovnání s analytickým řešením.
4. Odhad a postižení chyb.
5. PC implementace metody a řešení vybraných inženýrských úloh.

Seznam doporučené odborné literatury:

- Dubi A., Monte Carlo Applications in Systems Engineering, Wiley 2000, ISBN 0-471-981729.
- R.Briš, M.Litschmannová, Statistika I, elektronické skriptum VŠB TUO, FEI,2004

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **prof. Ing. Radim Briš, CSc.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. RNDr. Jiří Bouchala, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

V Ostravě 6. května 2013 Domesová .....

Ráda bych touto cestou poděkovala vedoucímu bakalářské práce prof. Ing. Radimu Brišovi, CSc. za věnovaný čas, cenné rady a poskytnutí podkladů pro vypracování této práce.

Abstrakt

Tato bakalářská práce je zaměřena na simulační metodu Monte Carlo, zejména na její aplikaci při modelování pohotovosti systémů s nezávislými prvky. Součástí práce je vytvoření programu v jazyce Matlab, který umožňuje modelování pohotovosti systémů s nezávislými prvky a řešení souvisejících úloh metodou Monte Carlo i analyticky. Důležitou část práce tvoří také srovnání analytické a simulační metody řešení a odhad chyby, k níž došlo při simulaci.

Klíčová slova: Monte Carlo, pohotovost, systém

Abstract

The bachelor thesis focuses on the Monte Carlo method, in particular on its application in availability modeling of systems with independent components. A part of the thesis is dedicated to a program implementation in Matlab, the program allows availability modeling of systems with independent components and solving related problems either by the Monte Carlo method or analytically. Another inportant part of this thesis is a comparison of analytical and simulation solving methods and estimation of the error caused by the simulation.

Keywords: Monte Carlo, availability, system

Seznam použitých zkratek a symbolů

CLV	–	Centrální limitní věta
CUDA	–	Compute Unified Device Architecture
GUI	–	Grafické uživatelské rozhraní (Graphical User Interface)
MC	–	Monte Carlo
MTTF	–	Střední doba provozu do poruchy (Mean Time To Failure)
MTTR	–	Střední doba do opravy (Mean Time To Repair)
NV	–	Náhodná veličina
PRSD	–	Procentuální relativní směrodatná odchylka (Percentage Relative Standard Deviation)
$\mathbb{R}$	–	Množina reálných čísel
TTF	–	Doba do poruchy (Time To Failure)
TTR	–	Doba do opravy (Time To Repair)

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tato práce je zaměřena na modelování pohotovosti systémů s nezávislými prvky pomocí simulační metody Monte Carlo.

Význam slova „systém“ v kontextu této práce bude vysvětlen dále, intuitivně si však lze představit jakýkoli soubor prvků, strojů, či jiných součástí, které spolupracují a dohromady vytváří jeden funkční celek, například přístroj z technické praxe, výrobní linku, počítač, či třeba lidské tělo.

Zaměřme se například na průmyslové systémy. Zájmem jejich provozovatele je docílit chodu systému s nejmenší možnou chybovostí, neboť v důsledku chyb je provoz systému přerušen a je třeba vyměnit či nahradit vadné prvky, což vede k finančním ztrátám. Proto je pro provozovatele takového systému výhodné znát jeho přibližnou spolehlivost ještě před jeho uvedením do provozu. Díky této simulaci budoucího chodu systému je schopen optimalizovat provozní náklady, určit, kolik zaměstnanců bude potřeba k údržbě systému, odhadnout předpokládané výnosy, apod.

K chybám systémových prvků může dojít z důvodu výrobních vad, lidského zavinění, obyčejného stárnutí materiálů, či zcela náhodně. Tato práce se zabývá právě systémy, v nichž k selháním dochází bez zjevných příčin.

Strukturu práce lze rozčlenit na teoretickou a praktickou část. Teorií se zabývají kapitoly 2 až 5. Ve 2. kapitole je shrnuta základní teorie z oblasti pravděpodobnosti, statistiky a Booleovy algebry. Další tři kapitoly se pak věnují hlavním tématům práce, tedy systémům, teorii spolehlivosti a metodě Monte Carlo. Jelikož je Monte Carlo metodou simulační, je odhadována také chyba, k níž při jejím použití dochází.

Kapitoly 6 až 8 se zabývají praktickou stránkou práce, tedy zejména aplikací metody Monte Carlo na problém modelování pohotovosti systémů s nezávislými prvky. Pro srovnání je tato úloha řešena také analyticky. Další důležitou součástí práce je implementace algoritmů pro modelování pohotovosti systému a vyřešení vybraných inženýrských úloh týkajících se této problematiky.

2 Základní teoretické poznatky

Tato kapitola si klade za cíl především stručné objasnění některých pojmů a základních teoretických poznatků z různých oborů. Jsou uvedeny pouze základy nutné pro pochopení dalšího textu, více je možné najít v odkazované literatuře.

Hlavním tématům (systémům, metodě Monte Carlo, teorii spolehlivosti) jsou pak věnovány samostatné kapitoly.

2.1 Teorie pravděpodobnosti

Podkladem pro tuto sekci je skriptum [3], kde lze najít podrobný výklad týkající se teorie pravděpodobnosti.

2.1.1 Pravděpodobnost

Náhodný pokus je každý děj, jehož výsledek není možné s jistotou předpovědět. Množina Ω všech možných vzájemně neslučitelných výsledků daného pokusu se nazývá **základní prostor**. **Náhodným jevem** je libovolná podmnožina A množiny Ω, **elementárním jevem** pak libovolná jednoprvková podmnožina ω množiny Ω.

Pravděpodobnost $P(A)$ označuje míru očekávatelnosti výskytu náhodného jevu A, pro její hodnotu platí $P(A) \in \langle 0,1 \rangle$.

Definice 2.1 (Klasická (Laplaceova) definice pravděpodobnosti) *Je-li základní prostor konečná neprázdná množina elementárních jevů* $\{\omega_1, \ldots, \omega_n\}$, *které mají stejnou šanci, tj. stejnou pravděpodobnost výskytu* $\left(\frac{1}{n}\right)$, *pak pravděpodobnost, že při realizaci náhodného pokusu jev A nastane, je*

$$P(A) = \frac{m}{n},$$

kde m je počet výsledků (elementárních jevů) příznivých jevu A a n počet všech možných výsledků.

Pro zpracování výsledků náhodného pokusu je třeba nejprve vytvořit model, který co nejlépe popisuje realitu, tímto modelem je tzv. náhodná veličina.

2.1.2 Náhodná veličina

Definice 2.2 (Náhodná veličina) *Náhodná veličina X (zkráceně NV X) je reálná funkce* $X : \Omega \to \mathbb{R}$ *taková, že pro každé reálné číslo x je množina*

$$\{\omega \in \Omega : X(\omega) < x\}$$

náhodným jevem.

Jsou rozlišovány dva základní typy náhodné veličiny, náhodná veličina s diskrétním rozdělením pravděpodobnosti (diskrétní NV) a se spojitým rozdělením pravděpodobnosti (spojitá NV). Diskrétní náhodná veličina nabývá pouze spočetně mnoha hodnot, zatímco spojitá může nabýt všech hodnot

z určitého intervalu, jejím příkladem je náhodně vybrané reálné číslo z intervalu $(0;1)$, doba do prasknutí žárovky, apod.

K popisu diskrétní NV se používá pravděpodobnostní a distribuční funkce.

Definice 2.3 (Pravděpodobnostní funkce) *Nechť diskrétní náhodná veličina X nabývá spočetně mnoha hodnot $\{x_1, x_2, \ldots\}$. Funkce $P(X = x_i) = P(x_i)$ se nazývá pravděpodobnostní funkce náhodné veličiny. Platí*

$$P(x_i) \geq 0 \quad a \quad \sum_{i=1}^{\infty} P(x_i) = 1.$$

Definice 2.4 (Distribuční funkce) *Nechť X je náhodná veličina. Reálnou funkci $F(x)$ definovanou pro všechna $x \in \mathbb{R}$ vztahem*

$$F(x) = P(X < x)$$

nazýváme distribuční funkcí náhodné veličiny X.

Distribuční funkce tedy každému reálnému číslu přiřazuje pravděpodobnost, že náhodná veličina nabude hodnoty menší než toto číslo.

K popisu spojité náhodné veličiny se kromě distribuční funkce používá hustota pravděpodobnosti. Pravděpodobnostní funkci nemá smysl pro spojitou NV definovat, pravděpodobnost, že spojitá NV nabude konkrétní hodnoty x, je totiž nulová pro všechna $x \in \mathbb{R}$.

Definice 2.5 (Hustota pravděpodobnosti) *Hustota pravděpodobnosti $f(x)$ spojité náhodné veličiny je reálná nezáporná funkce taková, že*

$$F(x) = \int_{-\infty}^{x} f(t)\,\mathrm{d}t$$

pro $-\infty < x < \infty$.

Pro modelování doby do výskytu události (např. doby do poruchy výrobku) je navíc používána tzv. hazardní funkce $\lambda(t)$. Nechť NV X představuje dobu do události. Je-li známo, že po dobu t nedošlo k události, pak $\lambda(t) \cdot \Delta t$ vyjadřuje pravděpodobnost, že k události dojde v následujícím krátkém časovém úseku délky Δt.

Definice 2.6 (Hazardní funkce) *Je-li X nezáporná náhodná veličina se spojitým rozdělením popsaným distribuční funkcí $F(t)$, definujeme pro $F(t) < 1$ hazardní funkci $\lambda(t)$ vztahem*

$$\lambda(t) = \frac{f(t)}{1 - F(t)}.$$

2.1.3 Číselné charakteristiky náhodné veličiny

Číselné charakteristiky náhodné veličiny X popisují vybrané vlastnosti této náhodné veličiny a používají se také pro srovnávání různých náhodných veličin. Dále jsou definovány nejdůležitější číselné charakteristiky, uvedené definiční vztahy platí, konverguje-li daná řada či integrál absolutně.

- **Obecný moment r-tého řádu** (pro $r \in \mathbb{N}$) se značí $E(X^r)$ nebo μ^r,

$$\mu^r = \sum_{(i)} x_i^r \cdot P(x_i) \text{ (diskrétní NV)}, \ \mu^r = \int_{-\infty}^{\infty} x^r \cdot f(x)\,\mathrm{d}x \text{ (spojitá NV)}.$$

- **Střední hodnota** se značí $E(X)$ nebo μ,

$$\mu = \sum_{(i)} x_i \cdot P(x_i) \text{ (diskrétní NV)}, \ \mu = \int_{-\infty}^{\infty} x \cdot f(x)\,\mathrm{d}x \text{ (spojitá NV)}.$$

- **Rozptyl** se značí $D(X)$ nebo σ^2,

$$\sigma^2 = \sum_{(i)} (x-\mu)^2 \cdot P(x_i) \text{ (diskrétní NV)}, \ \sigma^2 = \int_{-\infty}^{\infty} (x-\mu)^2 \cdot f(x)\,\mathrm{d}x \text{ (spojitá NV)}.$$

  Pro rozptyl rovněž platí $D(X) = E(X - E(X))^2 = E\left(X^2\right) - (E(X))^2$.

- **Směrodatná odchylka** σ je definována vztahem $\sigma = \sqrt{\sigma^2}$.

- **Kvantily** diskrétní NV se obvykle neurčují, pro spojitou NV je kvantil x_p, kde $p \in \langle 0;1 \rangle$, takové číslo, pro které platí

$$F(x_p) = p.$$

  Říkáme, že x_p je $100p\%$-ní kvantil dané náhodné veličiny.

2.1.4 Vybraná diskrétní rozdělení

Z diskrétních rozdělení pravděpodobnosti bude jmenováno rozdělení alternativní, binomické a Poissonovo. V souvislosti s nimi je třeba vysvětlit, co jsou to Bernoulliovy pokusy a Poissonův proces.

Uvažujme pokus, který má právě dva možné výsledky, s pravděpodobností p nastane úspěch a s pravděpodobností $1-p$ neúspěch. Náhodná veličina X popisující výsledek pokusu má **alternativní rozdělení**. Posloupnost takových nezávislých pokusů se označuje jako **Bernoulliovy pokusy**.

Má-li náhodná veličina X **binomické rozdělení**, píšeme $X \to Bi(n,p)$. Pravděpodobnostní funkce udává pravděpodobnost, že v n Bernoulliho pokusech dojde ke k úspěchům, platí

$$P(X=k) = \binom{n}{k} p^k (1-p)^{n-k}.$$

Poissonův proces popisuje počet náhodných událostí v pevném časovém intervalu délky t. Rychlost výskytu událostí (značí se λ) je v průběhu celého intervalu konstantní. Dále platí, že pravděpodobnost výskytu více než jedné události v limitně krátkém časovém intervalu je nulová. Počty událostí ve vzájemně disjunktních intervalech jsou nezávislé, pravděpodobnost výskytu události proto nezávisí na čase, který uplynul od minulé události.

Má-li náhodná veličina X **Poissonovo rozdělení**, píšeme $X \to Po(\lambda t)$. Pravděpodobnostní funkce udává pravděpodobnost, že v časovém intervalu délky t dojde ke k událostem, platí

$$P(X = k) = \frac{(\lambda t)^k \mathrm{e}^{-\lambda t}}{k!}.$$

2.1.5 Vybraná spojitá rozdělení

V této části jsou popsány vlastnosti vybraných spojitých rozdělení pravděpodobnosti, jež jsou zmiňována v dalším textu.

Rovnoměrné rozdělení je takové rozdělení, jehož hustota pravděpodobnosti je na nějakém intervalu $(a;b)$ konstantní a mimo tento interval nulová. Fakt, že náhodná veličina pochází z rovnoměrného rozdělení, se značí $X \to R(a;b)$. Pro hustotu pravděpodobnosti a distribuční funkci platí

$$\begin{aligned} f(x) &= \begin{cases} \frac{1}{b-a} & x \in (a;b) \\ 0 & x \notin (a;b), \end{cases} \\ F(x) &= \begin{cases} 0 & x \in (-\infty;a\rangle \\ \frac{x-a}{b-a} & x \in (a;b) \\ 1 & x \in \langle b;\infty). \end{cases} \end{aligned}$$

Exponenciální rozdělení se používá k popisu doby do výskytu události v Poissonově procesu a je tak hojně využíváno v teorii spolehlivosti. Zápis $X \to Exp(\lambda)$ značí, že má náhodná veličina X exponenciální rozdělení s parametrem λ. Pro hustotu pravděpodobnosti, distribuční a hazardní funkci platí

$$\begin{aligned} f(t) &= \begin{cases} \lambda \cdot e^{-\lambda t} & t > 0 \\ 0 & t \leqq 0, \end{cases} \\ F(t) &= \begin{cases} 1 - e^{-\lambda t} & t > 0 \\ 0 & t \leqq 0, \end{cases} \\ \lambda(t) &= \frac{f(t)}{1 - F(t)} = \frac{\lambda \cdot e^{-\lambda t}}{1 - 1 + e^{-\lambda t}} = \lambda. \end{aligned}$$

Díky tomu, že je hazardní funkce exponenciálního rozdělení konstantní, bývá tomuto rozdělení přezdíváno „rozdělení bez paměti“. Při modelování doby do selhání systému tímto rozdělením jsou si totiž následující dvě pravděpodobnosti:

- pravděpodobnost, že systém, který pracoval bez poruchy po dobu t_0, bude pracovat bez poruchy ještě alespoň po dobu t,

- pravděpodobnost, že systém, který dosud nebyl v provozu, bude pracovat bez poruchy alespoň po dobu t

rovny. Exponenciální rozdělení je tedy vhodné pro modelování chování systémů, u nichž dochází k poruše zcela náhodně, tj. ne v důsledku výrobních vad či opotřebení.

Erlangovo rozdělení popisuje dobu do výskytu k-té události v Poissonově procesu. Náhodou veličinu s tímto rozdělením lze tedy chápat jako součet k nezávislých náhodných veličin s exponenciálním rozdělením. Pro hustotu pravděpodobnosti platí vztah

$$f(t) = \begin{cases} \frac{\lambda(\lambda t)^{k-1} e^{-\lambda t}}{(k-1)!} & t > 0 \\ 0 & t \leq 0. \end{cases} \tag{2.1}$$

Zápis $X \to Erlang(k, \lambda)$ značí, že náhodná veličina X pochází z Erlangova rozdělení s parametry k a λ. [4]

Normální rozdělení je důležité zejména proto, že jím lze za určitých podmínek aproximovat řadu jiných rozdělení. Původ náhodné veličiny X z normálního rozdělení je označován zápisem $X \to N(\mu; \sigma^2)$, kde parametr μ (střední hodnota) charakterizuje polohu tohoto rozdělení a parametr σ^2 charakterizuje rozptýlení hodnot okolo střední hodnoty. Hustota pravděpodobnosti je dána výrazem

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

a jejím grafem je tzv. Gaussova křivka.

Jelikož hodnoty distribuční funkce normálního rozdělení nelze spočíst analyticky, je pro jejich určení využíván převod na tzv. **normované normální rozdělení** $N(0;1)$. Hustota pravděpodobnosti bývá obvykle značena $\varphi(z)$, platí

$$\varphi(z) = \frac{1}{\sqrt{2\pi}} \cdot \mathrm{e}^{-\frac{z^2}{2}}.$$

Distribuční funkce normovaného normálního rozdělení se značí $\phi(z)$ a její hodnoty v mnoha bodech byly vyčísleny pomocí numerických metod a tabelovány. Je-li X náhodná veličina s normálním rozdělením $N(\mu; \sigma^2)$, pro její distribuční funkci $F(x)$ platí převodní vztah

$$F(x) = \phi\left(\frac{x-\mu}{\sigma}\right).$$

Pro kvantily normovaného normálního rozdělení bude dále používáno označení z_p.

2.2 Statistika

Tato sekce vychází z [4]. Je zde popsán výběrový soubor a jeho charakteristiky, uvedeny některé limitní věty a shrnuto, čím se zabývá teorie odhadu.

2.2.1 Základní soubor vs. výběrový soubor

Množina všech prvků, které jsou sledovány při statistickém výzkumu, se nazývá **populace**, neboli **základní soubor**. Nejběžnějším statistickým výzkumem je tzv. **výběrové šetření**, kdy je zkoumána jen část populace, nazývaná **výběr**, neboli **výběrový soubor**. Závěry učiněné o výběru jsou poté induktivně přenášeny na celou populaci.

Výběrový soubor je obvykle získáván metodou **náhodného výběru**, v němž má každý prvek populace stejnou šanci na zařazení do výběrového souboru. Je-li n-krát opakován náhodný pokus, jehož výsledkem je hodnota náhodné veličiny X, a jsou-li tyto pokusy nezávislé, je tímto postupem získán náhodný výběr $X_1, \ldots, X_n$ z náhodné veličiny X o rozsahu n.

Údaj sledovaný u výběrového souboru se nazývá **proměnná** a její jednotlivé hodnoty **varianty** této proměnné. Proměnné podléhají následujícímu dělení:

- **Kategoriální proměnná** se nedá měřit, její varianty se dají pouze rozdělit do tříd a jsou vyjádřeny slovně.
- **Numerická proměnná** je měřitelná a její varianty jsou vyjádřeny číselně. Dělí se na diskrétní a spojitou.
  - **Diskrétní proměnná** nabývá konečného nebo spočetného množství variant.
  - **Spojitá proměnná** nabývá jakýchkoli hodnot z množiny $\mathbb{R}$ či její podmnožiny.

2.2.2 Některé charakteristiky výběrového souboru

Základními charakteristikami výběrového souboru jsou výběrový průměr a výběrový rozptyl. Výběrový průměr lze chápat jako výběrový protějšek střední hodnoty základního souboru, výběrový rozptyl pak jako protějšek rozptylu základního souboru.

Výběrový průměr je náhodná veličina definovaná vztahem

$$\bar{X} = \frac{1}{n} \cdot \sum_{i=1}^{n} X_i,$$

kde $X_1, \ldots, X_n$ je náhodný výběr a n jeho rozsah.

Výběrový rozptyl je mírou variability výběrového souboru, jedná se o náhodnou veličinu danou vztahem

$$s^2 = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (X_i - \bar{X})^2,$$

kde $X_1, \ldots, X_n$ je náhodný výběr a n jeho rozsah. **Výběrová směrodatná odchylka** s je pak dána jako $\sqrt{s^2}$.

2.2.3 Limitní věty

Definice 2.7 (Konvergence podle pravděpodobnosti) *Je dána posloupnost náhodných veličin $\{X_n\}$ a náhodná veličina X. Jestliže pro každé $\varepsilon > 0$ platí*

$$\lim_{n \to \infty} P(|X_n - X| < \varepsilon) = 1,$$

pak říkáme, že posloupnost náhodných veličin $\{X_n\}$ konverguje k náhodné veličině X podle pravděpodobnosti.

Věta 2.1 (Slabý zákon velkých čísel) *Nechť $X_1, X_2, \ldots$ je nekonečný náhodný výběr z rozdělení se střední hodnotou μ_X a konečným rozptylem σ_X^2, kde $X_1, X_2 \ldots$ jsou nekorelované náhodné veličiny. Potom výběrový průměr*

$$\bar{X}_n = \frac{1}{n} \cdot \sum_{i=1}^{n} X_i$$

vypočítaný z prvních n pozorování konverguje podle pravděpodobnosti k μ_X, tedy

$$\lim_{n\to\infty} P\left(|\bar{X}_n - \mu_X| < \varepsilon\right) = 1$$

pro každé $\varepsilon > 0$.

Věta 2.2 (Centrální limitní věta) *Jsou-li X_i nezávislé náhodné veličiny se stejným (libovolným) rozdělením a s konečným rozptylem, pak výběrový průměr má při dostatečně velkém počtu pozorování přibližně normální rozdělení.*

Píšeme

$$\bar{X} \sim N\left(\mu_X, \frac{\sigma_X^2}{n}\right) \quad \textit{nebo též} \quad \frac{\bar{X} - \mu_X}{\sigma_X}\sqrt{n} \sim N(0,1),$$

kde μ_X je střední hodnota a σ_X^2 rozptyl náhodné veličiny X.

2.2.4 Teorie odhadu

Teorie odhadu se zabývá odhadem parametrů populace na základě znalosti charakteristik výběrového souboru. Základními typy jsou bodový a intervalový odhad.

Bodový odhad je používán, je-li třeba parametr populace aproximovat konkrétním číslem, které je dále používáno ve výpočtech. Základními vlastnostmi dobrého odhadu T populačního parametru Θ jsou nestrannost, eficience a konzistence. Odhad je **nestranný**, pokud $E(T) = \Theta$. Nejlepším nestranným odhadem je odhad **eficientní**, tedy ten, který má nejmenší rozptyl ze všech nestranných odhadů parametru Θ. Pokud se odhad $T = T_n$ s rostoucím rozsahem výběru zpřesňuje (platí $\lim_{n\to\infty} E(T_n) = \Theta$, $\lim_{n\to\infty} D(T_n) = 0$), je tento odhad **konzistentní**.

Intervalový odhad je reprezentován intervalem spolehlivosti $\langle T_D, T_H \rangle$, v němž odhadovaný populační parametr Θ leží s pravděpodobností $1-\alpha$, platí

$$P(T_D \leq \Theta \leq T_H) = 1 - \alpha.$$

Číslo α se nazývá hladina významnosti, nejčastěji se volí $\alpha = 0,05$.

Je-li určován tzv. oboustranný interval spolehlivosti, je pravděpodobnost, že odhadovaný parametr leží pod dolní mezí T_D, rovna pravděpodobnosti, že leží nad horní mezí T_H, tedy

$$P(\Theta < T_D) = P(\Theta > T_H) = \frac{\alpha}{2}.$$

Tento interval se pak nazývá $100(1-\alpha)\,\%$ interval spolehlivosti pro parametr Θ.

2.3 Booleova algebra

Booleova algebra slouží pro práci s proměnnými, jež nabývají právě dvou hodnot, tyto hodnoty se označují symboly 0 a 1 nebo L a H a vyjadřují například stavy spínače (sepnut/rozepnut) nebo stavy systémového prvku (v provozu/mimo provoz).

Definice 2.8 (Logická proměnná) *Jestliže x je logická proměnná, která může nabývat pouze dvou hodnot (0,1), musí platit:*

$$x = 1, \text{ když } x \neq 0, \text{ a } x = 0, \text{ když } x \neq 1.$$

Definice 2.9 (Booleovská funkce) *Booleovská funkce $f(x_{n-1},\ldots,x_1,x_0)$ o n proměnných $x_{n-1},\ldots,x_1,x_0$ je zobrazení*

$$f : \{0,1\}^n \to \{0,1\},$$

kde $\{0,1\}^n$ je množina všech uspořádaných n-tic hodnot proměnných $x_{n-1},\ldots,x_1,x_0$.

Základními operátory Booleovy algebry jsou operátor logického součtu a operátor logického součinu. Jsou-li x_0 a x_1 logické proměnné, logický součet $x_0 + x_1$ je definován následovně:

$$x_0 + x_1 = 1 \Leftrightarrow (x_0 = 1 \vee x_1 = 1), \quad x_0 + x_1 = 0 \Leftrightarrow (x_0 = 0 \wedge x_1 = 0).$$

Pro logický součin $x_0 \cdot x_1$ platí

$$x_0 \cdot x_1 = 1 \Leftrightarrow (x_0 = 1 \wedge x_1 = 1), \quad x_0 \cdot x_1 = 0 \Leftrightarrow (x_0 = 0 \vee x_1 = 0).$$

Důsledkem je zákon idempotence, podle něhož

$$x_0 + x_0 = x_0, \quad x_0 \cdot x_0 = x_0.$$

Další zákony Booleovy algebry lze nalézt v [11].

3 Metoda Monte Carlo

Simulační metoda Monte Carlo je založena na teorii pravděpodobnosti a matematické statistice. Byla formulována matematiky Johnem von Neumannem a Stanislawem Ulamem během druhé světové války.

Matematické metody založené na statistickém výběru byly známy již odedávna, kvůli časové náročnosti výpočtů však nemohly být v praxi široce používány. V roce 1945 byl na univerzitě v Pensylvánii dokončen první elektronický počítač ENIAC, Stanislaw Ulam v této události viděl příležitost pro nové uplatnění statistických metod. Vědci Národní laboratoře Los Alamos ve Spojených státech amerických se v té době zabývali chováním neutronů, konkrétně bylo zkoumáno, do jaké vzdálenosti proniknou neutrony skrz různé materiály. Tento problém, jež byl tradičními deterministickými metodami neřešitelný, vyřešili Neumann a Ulam v roce 1947 právě pomocí metody Monte Carlo s využitím nových možností výpočetní techniky. [9]

Pomocí modelování náhodných veličin a následného statistického zpracování lze vyřešit řadu komplikovaných úloh, i takových, které s náhodou nemají nic společného. Díky rychlému vývoji výpočetní techniky našla metoda Monte Carlo uplatnění v širokém spektru přírodovědných, technických i ekonomických oborů. Například v matematice ji lze využít k řešení integrálů, zejména vícerozměrných, nebo k řešení systémů lineárních rovnic. [5]

3.1 Generování náhodných čísel

Každá aplikace metody Monte Carlo vyžaduje generování náhodných čísel, tj. generování hodnot náhodné veličiny s určitým rozdělením. Obvykle jsou nejprve generována náhodná čísla z rovnoměrného rozdělení $R(0;1)$, která jsou, má-li modelovaná náhodná veličina jiné rozdělení pravděpodobnosti, dále přepočítávána do požadovaného rozdělení.

Přirozeným zdrojem náhodných čísel jsou jsou hry založené na náhodě, například házení mincí, karetní a kostkové hry, ruleta. V hazardních hrách jsou tyto metody stále používány, nicméně pro počítačovou simulaci se nehodí, protože jsou příliš pomalé, navíc vygenerovanou posloupnost čísel není možné zopakovat. Některé moderní fyzikální generátory jsou již dostatečně rychlé, ale neopakovatelnost vygenerovaných sekvencí je stále nevýhodou.

Většina běžně používaných generátorů náhodných čísel je založena na počítačové implementaci jednoduchých rekurzivních algoritmů. Jelikož jsou tyto algoritmy deterministické, jsou takové generátory označovány jako generátory pseudonáhodných čísel. Příkladem je lineární kongruentní generátor, který pracuje na základě rekurentního předpisu

$$X_{i+1} = aX_i + c \pmod{m}$$

se zvolenou počáteční hodnotou X_0. [6]

3.2 Princip metody

Obecný postup řešení úlohy, jejímž přesným výsledkem je hodnota θ, pomocí metody Monte Carlo lze rozdělit do několika bodů:

1. Nejprve je třeba formulovat novou úlohu, jež má stochastický charakter a stejný výsledek jako původní úloha. Nová úloha spočívá v simulování původní úlohy pomocí náhodné veličiny X, jejíž střední hodnota je rovna θ.

2. Nová úloha je řešena pomocí mnohokrát opakovaných náhodných pokusů, jejichž výsledkem je hodnota náhodné veličiny X. V praxi jsou realizovány jako počítačem generovaná pseudonáhodná čísla, obvykle z rovnoměrného rozdělení $R(0;1)$, která jsou dále přepočítávána do rozdělení náhodné veličiny X. Je tedy získán náhodný výběr $X_1, X_2, \ldots X_n$ o rozsahu n, který je dále třeba statisticky zpracovat.

3. Je určen výsledek nové (stochastické) úlohy jako výběrový průměr $\bar{X}$ náhodného výběru $X_1, X_2, \ldots X_n$. Podle věty 2.1 výběrový průměr s rostoucím n konverguje ke střední hodnotě NV X, zde tedy k hodnotě θ.

4. Nakonec je třeba vyjádřit chybu řešení, tedy odchylku výsledku $\bar{X}$ stochastické úlohy od výsledku θ úlohy původní.

Jelikož však Monte Carlo není jednou konkrétní metodou, ale spíše souborem metod, může se postup v závislosti na aplikaci mírně lišit.

Při programové realizaci metody MC není třeba si pamatovat celý náhodný výběr (všech n hodnot). Je určován výběrový průměr $\bar{X}$, stačí tedy v cyklu spolu s generováním hodnot náhodné veličiny tyto hodnoty průběžně sčítat a konečný součet vydělit číslem n pro získání výběrového průměru. Podobně se postupuje i v případě určování rozptylu náhodného výběru, jak bude popsáno v následující sekci.

3.2.1 Určení chyby

Vzhledem k tomu, že se jedná o stochastickou metodu, nelze přesně určit chybu řešení, tzn. nelze vyčíslit odchylku $|\bar{X} - \theta|$. Namísto toho se určuje pravděpodobnost, že je tato odchylka menší než nějaké kladné číslo

$$\varepsilon = \alpha \cdot \frac{\sigma_X}{\sqrt{n}},$$

kde $\alpha \in \mathbb{R}^+$, σ_X je směrodatná odchylka náhodné veličiny X a n je rozsah náhodného výběru. [1]

S využitím Centrální limitní věty (Věta 2.2) lze psát

$$P(|\bar{X} - \mu_X| \leq \varepsilon) = P\left(|\bar{X} - \mu_X| \leq \alpha \cdot \frac{\sigma_X}{\sqrt{n}}\right) = P\left(-\alpha \leq \frac{\bar{X} - \mu_X}{\sigma_X}\sqrt{n} \leq \alpha\right) = \tag{3.1}$$

$$= \int_{-\alpha}^{\alpha} \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}x^2} \,\mathrm{d}x = \phi(\alpha) - \phi(-\alpha) = \phi(\alpha) - [1 - \phi(\alpha)] = 2\phi(\alpha) - 1,$$

kde μ_X značí střední hodnotu X, jež je ze zadání úlohy rovna θ, a $\phi(\alpha)$ je hodnota distribuční funkce normovaného normálního rozdělení v bodě α. Z (3.1) plyne, že bude-li se rozsah náhodného výběru n zvyšovat, bude se zvyšovat také přesnost odhadu.

Opačně lze najít číslo ε takové, že odchylka od přesného řešení je menší než ε s pravděpodobností $1-\alpha$. Z pohledu teorie odhadu je tedy určován $100\,(1-\alpha)\,\%$ interval spolehlivosti pro parametr $\bar{X}-\mu_X$ ve tvaru $\langle T_D;T_H\rangle = \langle -\varepsilon;\varepsilon\rangle$. Tento interval je symetrický okolo nuly díky symetrii normálního rozdělení, jímž je podle CLV náhodná veličina $\bar{X}$ aproximována, kolem střední hodnoty μ_X. Platí tedy

$$P\left(-\varepsilon \leq \bar{X}-\mu_X \leq \varepsilon\right) = P\left(|\bar{X}-\mu_X| \leq \varepsilon\right) = 2\phi\left(a\right)-1 = 1-\alpha \tag{3.2}$$

Jednoduchými úpravami lze vyjádřit neznámou ε.

$$\begin{aligned} 2\phi\left(a\right)-1 &= 1-\alpha \\ \phi\left(a\right) &= 1-\frac{\alpha}{2} \\ a=\varepsilon\cdot\frac{\sqrt{n}}{\sigma_X} &= z_{1-\frac{\alpha}{2}} \\ \varepsilon &= \frac{\sigma_X}{\sqrt{n}}\cdot z_{1-\frac{\alpha}{2}} \end{aligned} \tag{3.3}$$

Hodnotou ε, při níž je rovnost (3.2) splněna, je tedy $\frac{\sigma_X}{\sqrt{n}}\cdot z_{1-\frac{\alpha}{2}}$.

Ze vzorce (3.3) lze rovněž vyjádřit, jaký musí být rozsah výběrového souboru, aby pravděpodobnost, že je odchylka $|\bar{X}-\theta|$ menší nebo rovna jisté hodnotě $\varepsilon>0$, byla rovna číslu $1-\alpha$, tedy

$$n = \left(\frac{\sigma_X}{\varepsilon}\cdot z_{1-\frac{\alpha}{2}}\right)^2. \tag{3.4}$$

Směrodatná odchylka $\sigma_X = \sqrt{D\left(X\right)}$ obvykle není známa, je tedy třeba nahradit ji vhodným bodovým odhadem. Jako bodový odhad rozptylu se nabízí výběrový rozptyl s^2, je tedy třeba ověřit, zda je tento odhad nestranný, tedy jestli je střední hodnota výběrového rozptylu rovna rozptylu $D\left(X\right)$.

$$\begin{aligned} E\left(s^2\right) &= E\left(\frac{1}{n-1}\cdot\sum_{i=1}^{n}\left(X_i-\bar{X}\right)^2\right) = E\left(\frac{n}{n-1}\cdot\frac{1}{n}\sum_{i=1}^{n}\left(X_i-\bar{X}\right)^2\right) = \\ &= \frac{n}{n-1}\cdot E\left(\frac{1}{n}\sum_{i=1}^{n}X_i^2-\bar{X}^2\right) = \frac{1}{n-1}\cdot\sum_{i=1}^{n}E\left(X_i^2\right)-\frac{n}{n-1}\cdot E\left(\bar{X}^2\right) = \\ &= \frac{n}{n-1}\cdot\left(\sigma_X^2+\mu_X^2\right)-\frac{n}{n-1}\cdot\left(\frac{\sigma_X^2}{n}+\mu_X^2\right) = \sigma_X^2 \end{aligned}$$

Nestrannost výběrového rozptylu s^2 jako odhadu skutečného rozptylu $D\left(X\right)$ byla potvrzena, směrodatnou odchylku σ_X tedy lze odhadnout výběrovou směrodatnou odchylkou s.

Další používanou mírou přesnosti odhadu $\bar{X}$ je procentuální relativní směrodatná odchylka *PRSD* (z anglického *Percentage Relative Standard Deviation*), definovaná vztahem

$$PRSD = 100\cdot\frac{s}{\bar{X}\cdot\sqrt{n}}.$$

Pro $n\to\infty$ konverguje *PRSD* k relativní chybě odhadu $\bar{X}$, pro níž platí

$$V_{\bar{X}} = \frac{\sqrt{D\left(\bar{X}\right)}}{E\left(\bar{X}\right)} = \frac{\sqrt{\frac{D(X)}{n}}}{E\left(X\right)} = \frac{\sigma_X}{\mu_X\cdot\sqrt{n}}$$

a bývá také nazývána variační koeficient náhodné veličiny $\bar{X}$.

Jak v případě hledání intervalu spolehlivosti pro parametr $\bar{X} - \mu_X$ (odchylku od přesného řešení), tak i v případě určování *PRSD*, je k výpočtu nutná znalost výběrové směrodatné odchylky s. Při řešení úlohy metodou MC je tedy potřeba počítat nejen výběrový průměr, ale i výběrový rozptyl s^2, pomocí něhož lze výběrovou směrodatnou odchylku spočíst. Platí

$$\begin{aligned} s^2 &= \frac{1}{n-1} \cdot \sum_{i=1}^{n} (X_i - \bar{X})^2 = \frac{1}{n-1} \cdot \sum_{i=1}^{n} \left(X_i^2 - 2X_i\bar{X} + \bar{X}^2 \right) = \\ &= \frac{1}{n-1} \cdot \left(\sum_{i=1}^{n} X_i^2 - 2n\bar{X}\bar{X} + n\bar{X}^2 \right) = \frac{1}{n-1} \cdot \left(\sum_{i=1}^{n} X_i^2 - n\bar{X}^2 \right), \end{aligned}$$

pro výpočet s tedy vedle střední hodnoty $\bar{X}$ stačí znát již pouze sumu druhých mocnin hodnot X_i, kterou je možné počítat v cyklu spolu se sumou hodnot X_i.

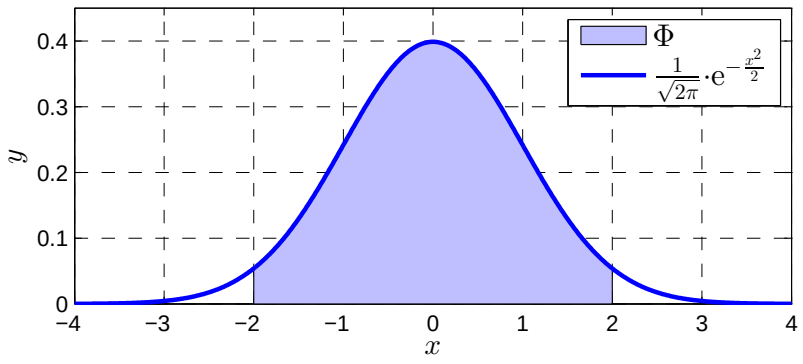
3.3 Příklad použití metody Monte Carlo

Výše uvedený postup bude nyní znázorněn při výpočtu určitého integrálu funkce jedné proměnné. Nejjednodušším způsobem výpočtu integrálu pomocí metody Monte Carlo je využití geometrické interpretace integrálu. Je-li funkce $f(x)$ nezáporná na intervalu $\langle a;b \rangle$, určitý integrál od a do b z funkce $f(x)$ je roven obsahu plochy ohraničené grafem funkce $f(x)$, osou x a přímkami $x = a$ a $x = b$.

Příklad 3.1
Úkolem je odhadnout hodnotu určitého integrálu

$$\int_{-2}^{2} \frac{1}{\sqrt{2\pi}} \cdot \mathrm{e}^{-\frac{x^2}{2}} \, \mathrm{d}x$$

a přesnost tohoto odhadu.



Obrázek 3.1: Geometrická interpretace integrálu z příkladu 3.1

Podle geometrické interpretace integrálu je počítán obsah oblasti Φ znázorněné na obrázku 3.1. Tuto oblast je třeba ohraničit vhodnou, nejlépe co nejmenší, obdélníkovou oblastí Ω. Abychom tak

mohli učinit, nejprve najdeme maximum integrované funkce na intervalu $\langle -2;2\rangle$. Snadno zjistíme, že má maximum v bodě $x=0$ a pro jeho hodnotu platí $f(0)=\frac{1}{\sqrt{2\pi}}<0,4$. Jako ohraničující oblast Ω tedy zvolíme obdélník $\langle -2;2\rangle \times \langle 0;0,4\rangle$. Pro obsah S oblasti Ω platí $S=1,6$.

Nechť $[x,y]$ je náhodný bod z oblasti Ω. Označíme-li symbolem I obsah oblasti Φ, pro pravděpodobnost, že náhodný bod $[x,y]$ leží v oblasti Φ, platí

$$P([x,y]\in\Phi)=\frac{I}{S}.$$

Pro hledaný obsah I tedy platí

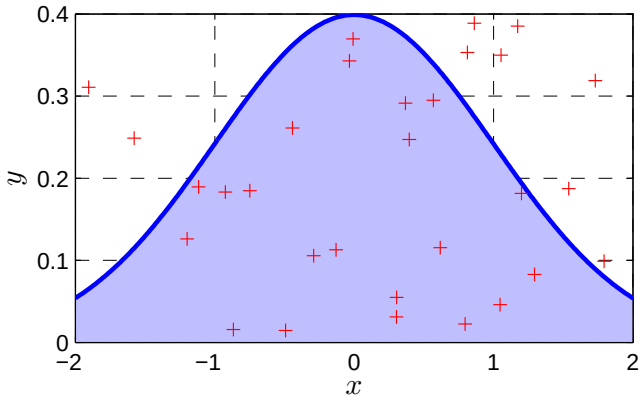
$$I=S\cdot P([x,y]\in\Phi).$$

Dále postupujme podle bodů uvedených v sekci 3.2.

1. Náhodnou veličinou X je ohodnocení náhodného bodu $[x,y]$ z oblasti Ω v závislosti na tom, zda leží v oblasti Φ či nikoliv, tedy

$$X=\begin{cases} S & \text{když } [x,y]\in\Phi \\ 0 & \text{když } [x,y]\notin\Phi \end{cases}. \tag{3.5}$$

2. Náhodný výběr spočívá v generování n náhodných bodů $[x_i,y_i]$ z oblasti Ω a určení, zda platí $X_i=0$ nebo $X_i=S$ podle (3.5), viz obrázek 3.2.



Obrázek 3.2: Generování náhodných bodů z oblasti Ω z příkladu 3.1

3. Hledanou hodnotou integrálu, resp. obsahu I, je střední hodnota náhodné veličiny X, kterou odhadneme výběrovým průměrem $\bar{X}$ náhodného výběru $X_1,X_2,\ldots,X_n$, tedy

$$I\approx\bar{X}=\frac{1}{n}\cdot\sum_{1}^{n}X_i. \tag{3.6}$$

(Označíme-li n_{in} počet vygenerovaných bodů, jež padly do oblasti Φ, můžeme rovněž psát

$$I\approx\bar{X}=S\cdot\frac{n_{in}}{n},$$

tento zápis je ekvivalentní s (3.6) a je vhodnější pro implementaci.)

4. Odchylku ε od přesného řešení určíme podle (3.3). Za hladinu významnosti α dosadíme $\alpha = 0.05$ a za n rozsah realizovaného náhodného výběru. Jelikož směrodatná odchylka σ_X není známa, nahradíme ji výběrovou směrodatnou odchylkou s náhodného výběru $X_1, X_2, \ldots, X_n$. Jelikož náhodná veličina X nabývá pouze dvou hodnot, z nichž jedna je nulová, lze vzorec pro výpočet s upravit tímto způsobem:

$$s = \sqrt{\frac{1}{n-1} \cdot \left(\sum_{i=1}^{n} X_i^2 - n\bar{X}^2 \right)} = \sqrt{\frac{1}{n-1} \cdot (n_{in} S^2 - n\bar{X}^2)}.$$

   Není tedy nutné počítat průběžně sumu X_i^2, stačí dosadit n, n_{in}, S a $\bar{X}$.

Realizaci úlohy provedeme implementací v jazyce Matlab, viz výpis 3.1.

```
function [ I, epsilon, PRSD ] = priklad_3_1( a, b, n, alpha )
%   a, b    integrační meze
%   n       rozsah náhodného výběru
S=(b-a)*0.4;              % obsah obdélníkové oblasti Omega
n_in=0;                   % počet bodů, které padly pod křivku
for i=1:n
    x=rand()*(b-a)+a;     % generování náhodných hodnot z R(a;b)
    y=rand()*0.4;         % generování náhodných hodnot z R(0;0.4)
    % rozhodnutí, zda [x,y] leží pod křivkou
    if y < exp(-(x^2)/2)/sqrt(2*pi)
        n_in=n_in+1;      % navýšení počtu náh. bodů, které padly pod křivku
    end
end
I=S*n_in/n;                           % výběrový průměr
s=sqrt((n_in*S^2-n*I^2)/(n-1));       % výběrová směrodatná odchylka
epsilon=(s*norminv(1-alpha/2))/sqrt(n);
PRSD=100*s/(I*sqrt(n));
```

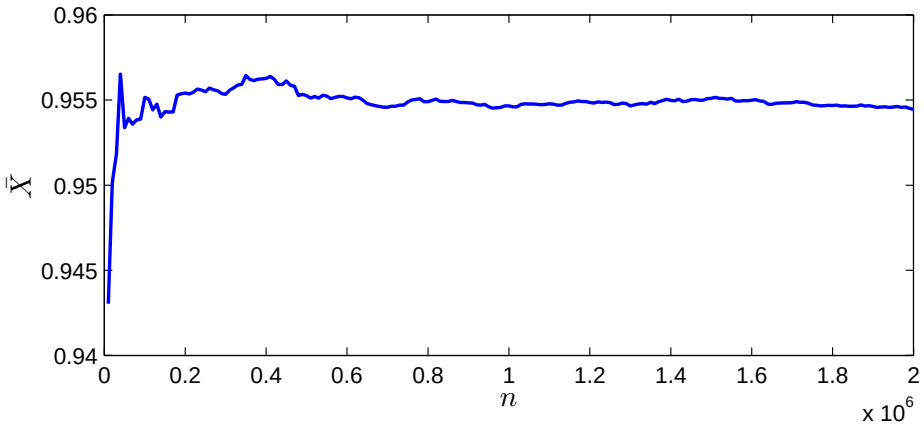
Výpis 3.1: Implementace příkladu 3.1 v jazyce Matlab

Výsledky úlohy (s přesností na čtyři desetinná místa) v závislosti na počtu provedených náhodných pokusů jsou zaznamenány v následující tabulce. Je patrné, že se výpočetní čas s rostoucím n zvyšuje přibližně lineárně. Hodnoty ε a *PRSD* dle očekávání s rostoucím n klesají, přesnost odhadu se tedy zvyšuje.

Rozsah náhodného výběru (n)	10^2	10^3	10^4	10^5	10^6	10^7
Odhad hodnoty integrálu (I)	0.9120	0.9472	0.9430	0.9552	0.9547	0.9544
Výběrová směrodatná odchylka	0.7961	0.7867	0.7871	0.7848	0.7849	0.7850
Hodnota ε při $\alpha = 0.05$	0.1560	0.0488	0.0154	0.0049	0.0015	0.0005
Skutečná chyba odhadu $\lvert I - \bar{X} \rvert$	0.0425	0.0073	0.0115	0.0007	0.0002	0.0001
PRSD	8.73%	2.63%	0.83%	0.26%	0.08%	0.03%
Čas výpočtu v sekundách	0.0004	0.0020	0.0174	0.1718	1.6955	16.9145

Tabulka 3.1: Výsledky úlohy 3.1

Proces zpřesňování odhadu a konvergenci k hodnotě I s rostoucím rozsahem náhodného výběru (pro n od 10^4 do $2 \cdot 10^6$) zachycuje také graf na obrázku 3.3.



Obrázek 3.3: Výsledek příkladu 3.1 v závislosti na rozsahu náhodného výběru

Výsledkem úlohy, tedy odhadem tohoto určitého integrálu získaným metodou MC, je hodnota 0.9544. S pravděpodobností 0.95 je chyba tohoto odhadu nižší než $5 \cdot 10^{-4}$, viz poslední sloupec tabulky 3.1.

△

Poznámka 3.1 Funkce z příkladu 3.1 je hustotou pravděpodobnosti normovaného normálního rozdělení. Řešený integrál lze tedy vyjádřit také jako

$$I = \phi(2) - \phi(-2) \doteq 0.9544997.$$

Jelikož jsou hodnoty distribuční funkce $\phi(z)$ normovaného normálního rozdělení tabelovány, bylo možné určit hodnotu I a porovnávat ji s odhadem $\bar{X}$, přestože primitivní funkci k $\frac{1}{\sqrt{2\pi}} \cdot \mathrm{e}^{-\frac{x^2}{2}}$ nelze vyjádřit pomocí konečného počtu elementárních funkcí.

4 Systémy a jejich reprezentace

Úkolem kapitoly je objasnit, co v kontextu této práce znamená pojem systém a související pojmy komponenta, stavový indikátor, stavový vektor, stavový prostor a systémová funkce. Vychází především z [1].

4.1 Komponenta

Definice 4.1 (Komponenta) *Jednotka se nazývá komponentou, jestliže má následující vlastnosti:*

1. *Má konečný počet diskrétních známých stavů.*
2. *Je znám mechanismus přechodu mezi jejími stavy.*

Komponenta je určená konečnou množinou diskrétních stavů, jichž může nabýt. Stav komponenty je označen proměnnou b, která se nazývá stavový indikátor. V určitém čase t se komponenta nachází v právě jenom z těchto stavů. Tato práce se zabývá komponentami, které mají právě dva možné stavy:

Stav 0 Jednotka je mimo provoz, závada však byla detekována a probíhá oprava. Po dokončení opravy bude jednotka opět v provozu. Platí $b = 0$.

Stav 1 Jednotka je v provozu, tedy v operativním stavu. Platí $b = 1$.

Obecně však komponenta může nabývat širšího spektra stavů. Jedná se například o pasivní stav, kdy nedošlo k závadě, ale jednotka je uvedena mimo provoz, stav, kdy byla detekována závada, prozatím však není možné zahájit opravu, stav zátěže, ve kterém je jednotka náchylnější k opotřebení, a o mnoho dalších.

Historie komponenty může být popsána jako posloupnost přechodů mezi jednotlivými stavy. Přechody mezi stavy mohou mít stochastickou i deterministickou povahu. Deterministickou změnou stavu je například plánované uvedení komponenty mimo provoz z důvodu údržby. Z pohledu problematiky této práce jsou však důležité přechody stochastické, ty se dále dělí na přechody způsobené vzájemným ovlivňováním jednotlivých komponent a přechody způsobené vlastním náhodným procesem jedné komponenty.

Z důvodu zjednodušení jsou dále uvažovány pouze posledně jmenované přechody, tedy takové, které se týkají jediné komponenty. Příkladem náhodného jevu, který způsobí změnu stavu komponenty, může být například náhlé prasknutí žárovky.

4.2 Systém

Systém je souborem n komponent, $n \in \mathbb{N}$. Každá z komponent je opatřena stavovým indikátorem, určujícím, ve kterém z možných stavů se daná komponenta nachází, i-tá komponenta má stavový indikátor b_i. Vektor stavových indikátorů se nazývá stavovým vektorem systému a značí se $\mathbf{B}$, platí $\mathbf{B} = (b_1, b_2, \ldots b_n)$. Množina všech možných stavových vektorů se nazývá stavovým prostorem systému a značí se V. Jelikož jsou uvažovány pouze komponenty mající právě dva možné stavy, velikost stavového prostoru V je 2^n.

Stav systému v čase t je jednoznačně určen stavy jednotlivých komponent v čase t, tedy příslušným stavovým vektorem systému. V nejjednodušším případě má systém právě dva možné stavy, v provozu a mimo provoz. Závislost stavu systému na stavech jeho komponent popisuje systémová funkce.

Systém je možné chápat také jako stavový vektor měnící se v čase. Změna stavového vektoru je důsledkem změny stavu některé z komponent. Dále jsou uvažovány pouze systémy, jejichž komponenty se navzájem neovlivňují, systémem je tedy dále myšlen systém s nezávislými komponentami.

4.2.1 Systémová funkce

Jedná se o funkci, která každému stavu systému přiřazuje právě jedno reálné číslo. Definičním oborem systémové funkce je tedy stavový prostor systému, oborem hodnot množina reálných čísel či její podmnožina. K jednomu systému je možné přiřadit více funkcí, každá z nich může vypovídat o jiné kvalitě systému.

Příklad 4.1
Uvažujme systém o dvou komponentách reprezentujících potrubí s proudící kapalinou. První komponenta má průtok 0.3 $\mathrm{m^3 s^{-1}}$, druhá 0.2 $\mathrm{m^3 s^{-1}}$. Stavový indikátor i-té komponenty b_i je roven číslu 1, je-li komponenta v provozu, a číslu 0, je-li mimo provoz.

Systémová funkce vyjadřující průtok systémem má tvar

$$S(\mathbf{B}) = 0.3b_1 + 0.2b_2.$$

△

Příklad 4.2
Uvažujme systém z předchozího příkladu.

Najděme nyní systémovou funkci, která vrací číslo 1, je-li systém v provozu (je v provozu alespoň jedna z komponent), a číslo 0, je-li mimo provoz (průtok systémem je nulový). Tuto funkci můžeme zapsat například následujícím způsobem:

$$S(\mathbf{B}) = \begin{cases} 1 & \text{jestliže } b_1 + b_2 > 0 \\ 0 & \text{jestliže } b_1 + b_2 = 0. \end{cases}$$

K jednomu systému tedy zřejmě je možné přiřadit více různých systémových funkcí.

△

Poznámka 4.1 Systémovou funkci z příkladu (4.2) je možné považovat za funkci logických proměnných. Přejde tak do tvaru

$$S(\mathbf{B}) = b_1 + b_2$$

a bude stále vracet hodnotu 1 v případě funkčnosti systému a hodnotu 0 v opačném případě.

Nyní je již možné přistoupit k definici systému.

Definice 4.2 (Systém) *Systém je souborem komponent, na němž je definována alespoň jedna systémová funkce.*

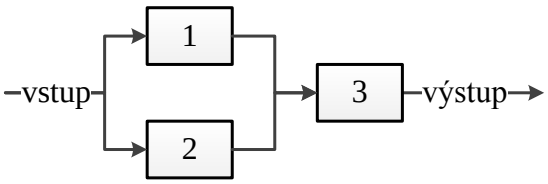
4.2.2 Reprezentace systému pomocí schématu

Tato sekce se věnuje hledání systémové funkce v případech, kdy je k dispozici schéma popisující strukturu systému. V literatuře bývá označováno pojmy *funkční schéma*, nebo také *blokové schéma*. Jedná se o orientovaný graf mající dva speciální vrcholy (dále ve schématech označovány jako „vstup“ a „výstup“), zbylé vrcholy reprezentují komponenty systému. Orientovaná hrana spojující dva vrcholy značí, že existuje přímá cesta mezi příslušnými systémovými prvky ve směru vyznačeném šipkou.

Je třeba zdůraznit, že toto schéma popisuje logickou strukturu systému, která nemusí nijak souviset se strukturou fyzickou. Má pouze přehledně znázornit, jaké kombinace komponent musí být v provozu, aby byl v provozu celý systém, a usnadnit tak hledání příslušné systémové funkce.

Hledanou systémovou funkcí je zde funkce definovaná na stavovém prostoru systému, která vrací hodnotu 0 nebo 1 v závislosti na stavu systému. Systém je v provozu (ve stavu 1), existuje-li cesta ze vstupu na výstup pouze přes funkční komponenty, v opačném případě je systém mimo provoz. Práce se dále zabývá především tímto typem systémové funkce, proto již není třeba systémovou funkci považovat za reálnou funkci reálných proměnných. Nebude-li uvedeno jinak, bude systémová funkce dále chápána jako booleovská funkce logických proměnných, což povede ke zjednodušením a ke zpřehlednění zápisu.

Na konkrétním příkladu budou nyní vysvětleny výše zmiňované pojmy a ukázán postup nalezení systémové funkce podle schématu popisujícího logickou strukturu (dále jen strukturu) systému.



Obrázek 4.1: Schéma systému z příkladu 4.3

Příklad 4.3
Je dán systém o třech komponentách. Jeho struktura je znázorněna na obrázku 4.1. Je-li i-tá komponenta v provozu, platí $b_i = 1$, je-li mimo provoz, platí $b_i = 0$. Stavovým prostorem systému je tedy tato množina vektorů:

$$V = \{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0),(1,1,1)\},$$

obsahující všechny možné kombinace stavů jednotlivých komponent, tj. všechny možné stavy systému.

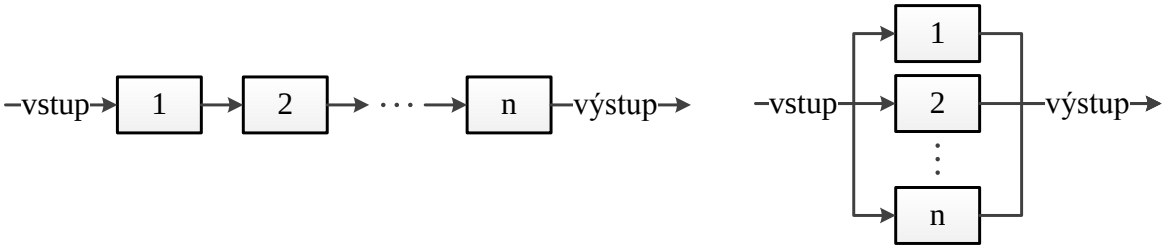
Nyní je třeba nalézt systémovou funkci, která bude vracet hodnotu 1, je-li systém v provozu, a hodnotu 0, je-li mimo provoz. K tomu poslouží obrázek 4.1. Systém je v provozu, pokud je možné najít cestu ze vstupu na výstup pouze přes komponenty, které jsou v provozu. Musí tedy být v provozu komponenta č. 3 a alespoň jedna z komponent č. 1 a 2, tomu odpovídají stavové vektory $(0,1,1)$, $(1,0,1)$ a $(1,1,1)$.

Označme $V_1 = \{(0,1,1),(1,0,1),(1,1,1)\}$ množinu vektorů označujících stavy, v nichž je systém v provozu a $V_0 = V \setminus V_1 = \{(0,0,0),(0,0,1),(0,1,0),(1,0,0),(1,1,0)\}$ množinu stavových vektorů, pro které je systém mimo provoz. Jednoduše ukážeme, že hledanou systémovou funkcí je

$$S(\mathbf{B}) = (b_1 + b_2) \cdot b_3,$$

stačí ověřit, že $\forall \mathbf{B} \in V_0 : S(\mathbf{B}) = 0$ a $\forall \mathbf{B} \in V_1 : S(\mathbf{B}) = 1$. △

Základními strukturami systému jsou čistě sériová a čistě paralelní struktura. V těchto případech je nalezení systémové funkce snadné.



(a) Sériová struktura (b) Paralelní struktura

Obrázek 4.2: Schémata základních logických struktur systémů

Příklad 4.4 (Sériová struktura)
Je-li systém složen z n komponent spojených sériově (viz obrázek 4.2a), je v provozu pouze tehdy, jsou-li v provozu všechny komponenty. Systémová funkce má tvar

$$S(\mathbf{B}) = b_1 \cdot b_2 \cdot \ldots \cdot b_n.$$

△

Příklad 4.5 (Paralelní struktura)
Má-li systém n komponent paralelní strukturu (viz obrázek 4.2b), je v provozu právě tehdy, když je v provozu alespoň jedna komponenta. Systémová funkce má tvar

$$S(\mathbf{B}) = 1 - (1 - b_1) \cdot (1 - b_2) \cdot \ldots \cdot (1 - b_n).$$

Lze snadno ověřit, že má-li alespoň jeden ze stavových indikátorů hodnotu 1, platí $S(\mathbf{B}) = 1$.

△

O něco složitější strukturou je sério-paralelní kombinace. Již z názvu je však patrné, že systémovou funkci je možné nalézt zkombinováním postupů používaných u čistě paralelní a čistě sériové struktury. Jednoduchým případem sério-paralelní kombinace je i systém z příkladu 4.3.

Jiné systémy, jež nemají sério-paralelní strukturu, je možné na tuto strukturu převést. K tomu se používají například metody využívající hledání minimálních řezů a minimálních drah (anglicky *minimal cut set*, *minimal tie set*).

4.2.3 Metoda minimálních drah, metoda minimálních řezů

Tyto metody slouží ke hledání systémových funkcí systémů, jež nemají sério-paralelní strukturu, nebo je tato struktura příliš složitá. Před jejich vysvětlením je třeba nejprve definovat některé pojmy.

Dráha je množina komponent, pro které platí, že jsou-li všechny v provozu, je v provozu celý systém. **Minimální dráhou** je taková množina komponent, která je dráhou a pro níž platí, že po odebrání libovolného prvku již dráhou není.

Řez je množina komponent, pro něž platí, že jsou-li všechny mimo provoz, je nutně mimo provoz celý systém. **Minimálním řezem** je každá množina komponent, která je řezem a která po odebrání libovolného prvku již dále řezem není.

Nechť je dáno schéma popisující strukturu systému. Postup nalezení systémové funkce pomocí metody minimálních drah je následující:

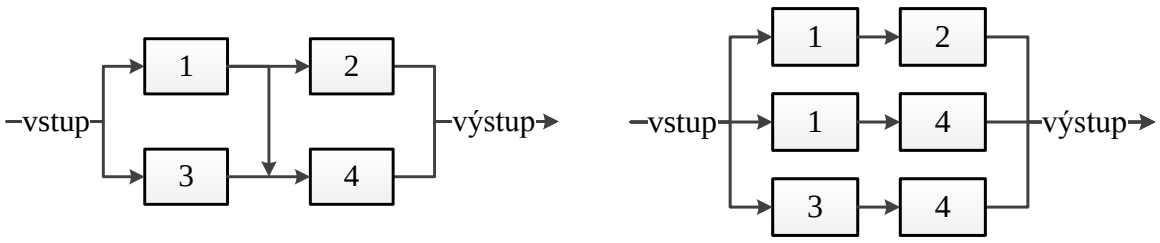
1. Nejprve je třeba nalézt všechny minimální dráhy a sestavit alternativní schéma s ekvivalentní strukturou. Komponenty tvořící minimální dráhu jsou spojeny do série, jednotlivé minimální dráhy jsou poté propojeny paralelně.

2. Dalším krokem je zapsání systémové funkce podle postupů používaných v příkladech 4.4 a 4.5.

3. Nakonec je třeba roznásobit systémovou funkci a upravit podle pravidel Booleovy algebry, tedy především zredukovat mocniny stavových indikátorů na jedničky.

Postup pomocí metody minimálních řezů je obdobný. Liší se pouze v bodě 1, kdy jsou určovány minimální řezy, jejichž komponenty jsou propojeny paralelně, jednotlivé minimální řezy jsou poté spojeny do série. Výsledkem obou metod je vždy stejný tvar systémové funkce, nazývá se **normalizovaný tvar systémové funkce**.

Příklad 4.6 (Obecná struktura)
Příkladem systému, jehož struktura není sério-paralelní, je systém na obrázku 4.3a. Pro nalezení odpovídající systémové funkce je nutné tento graf systému převést na ekvivalentní sério-paralelní kombinaci. K tomu využijeme minimální dráhy, jsou jimi množiny $\{1,2\}$, $\{1,4\}$ a $\{3,4\}$. Jednotlivé prvky v množinách spojíme do série, množiny navzájem propojíme paralelně, viz obrázek 4.3b.



(a) Obecná struktura (b) Ekvivalentní sério-paralelní struktura

Obrázek 4.3: Systém z příkladu 4.6

Na graf budeme pohlížet jako na paralelní spojení skupin sériově propojených komponent a při hledání systémové funkce využijeme poznatků z příkladů 4.4 a 4.5. Při běžném značení tedy získáme

funkci

$$S(\mathbf{B}) = 1 - (1 - b_1 b_2) \cdot (1 - b_1 b_4) \cdot (1 - b_3 b_4),$$

kterou dále upravíme na tvar

$$S(\mathbf{B}) = b_1 b_2 + b_1 b_4 + b_3 b_4 - b_1 b_3 b_4^2 - b_1^2 b_2 b_4 - b_1 b_2 b_3 b_4 + b_1^2 b_2 b_3 b_4^2.$$

Jelikož $S(\mathbf{B})$ je funkce logických proměnných, pro něž platí $b_i^k = b_i$, kde $i,k \in \mathbb{N}$, lze všechny druhé mocniny zredukovat na první. Funkce tedy přejde do tvaru

$$\begin{aligned} S(\mathbf{B}) &= b_1 b_2 + b_1 b_4 + b_3 b_4 - b_1 b_3 b_4 - b_1 b_2 b_4 - b_1 b_2 b_3 b_4 + b_1 b_2 b_3 b_4 = \\ &= b_1 b_2 + b_1 b_4 + b_3 b_4 - b_1 b_3 b_4 - b_1 b_2 b_4. \end{aligned} \tag{4.1}$$

Podobně lze úlohu řešit metodou minimálních řezů. Těmi jsou množiny $\{1,3\}$, $\{1,4\}$ a $\{2,4\}$. Zapíšeme systémovou funkci

$$S(\mathbf{B}) = [1 - (1 - b_1) \cdot (1 - b_3)] \cdot [1 - (1 - b_1) \cdot (1 - b_4)] \cdot [1 - (1 - b_2) \cdot (1 - b_4)].$$

Po roznásobení a úpravě přejde funkce opět do normalizovaného tvaru (4.1).

△

Poznámka 4.2 Má-li systémová funkce normalizovaný tvar, je možné stavové indikátory b_i nahradit pravděpodobnostmi p_i, že je i-tá komponenta v jistém čase v provozu. Výsledkem systémové funkce pak není pouze stav systému (hodnota 0 nebo 1), ale přímo pravděpodobnost, že je systém v uvažovaném čase v provozu. Tato pravděpodobnost bude dále definována jako pohotovost.

Příklad 4.7
Uvažujme opět systém z příkladu (4.6). Jeho normalizovanou systémovou funkci již známe, jedná se o funkci

$$S(\mathbf{B}) = b_1 b_2 + b_1 b_4 + b_3 b_4 - b_1 b_3 b_4 - b_1 b_2 b_4.$$

Určeme nyní pravděpodobnost, že je systém v jistém čase v provozu, víme-li, že pro pravděpodobnosti p_i, že je v tomto čase v provozu i-tá komponenta, platí

$$p_1 = 0,95,\ p_2 = 0,9,\ p_3 = 0,85,\ p_4 = 0,8.$$

Jak již bylo řečeno, pro tento účel je třeba stavové indikátory b_i nahradit pravděpodobnostmi p_i. Do upravené funkce poté dosadíme konkrétní hodnoty pravděpodobností, tedy

$$\begin{aligned} S(\mathbf{B}) &= p_1 p_2 + p_1 p_4 + p_3 p_4 - p_1 p_3 p_4 - p_1 p_2 p_4 = \\ &= 0,95 \cdot 0,9 + 0,95 \cdot 0,8 + 0,85 \cdot 0,8 - 0,95 \cdot 0,85 \cdot 0,8 - 0,95 \cdot 0,9 \cdot 0,8 = \\ &= 0,8795. \end{aligned}$$

Pravděpodobnost, že je systém v uvažovaném čase v provozu, je $0,8795$.

△

Je však nutné si uvědomit, že hledání normalizované systémové funkce je náročné, v případě obecných systémů o stovkách komponent až nemožné. Proto je třeba najít univerzálnější způsob, jak vypočíst pohotovost systému v případě, kdy normalizovaná systémová funkce není k dispozici.

5 Základy teorie spolehlivosti

Úkolem této kapitoly je definování některých pojmů z teorie spolehlivosti. Zejména je třeba objasnit, co znamená pohotovost systému, jejíž kvantifikace je hlavním cílem této bakalářské práce. Kapitola čerpá z [1] a další dále odkazované literatury.

Pojmy budou vysvětlovány na jedné komponentě, nebo přesněji na systému tvořeném jedinou komponentou, která má právě dva možné stavy, 0 a 1. Přechod ze stavu 1 do stavu 0 bude dále nazýván poruchou, přechod ze stavu 0 do stavu 1 opravou.

5.1 Doba do poruchy, doba do opravy

Předpokládejme, že je systém v čase $t = 0$ v provozu. K první poruše dojde v čase X. Doba X, po kterou byl systém v provozu, se nazývá **doba do poruchy**. Doba do poruchy je dále považována za náhodnou veličinu, pro níž platí:

- Distribuční funkce se značí $F_1(t)$ a vyjadřuje pravděpodobnost, že na intervalu $(0,t)$ dojde k poruše.
- Hustota pravděpodobnosti se značí $f_1(t)$.
- Střední hodnota se nazývá **střední doba provozu do poruchy** a je označována zkratkou MTTF (z anglického *mean time to failure*).
- Hazardní funkce se nazývá **intenzita poruch**.

Analogicky lze definovat náhodnou veličinu **doba do opravy**, platí pro ni:

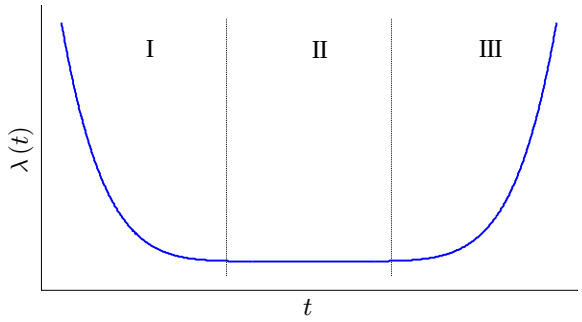
- Distribuční funkce $F_0(t)$ vyjadřuje pravděpodobnost, že na intervalu $(0,t)$ dojde k opravě.
- Hustota pravděpodobnosti se značí $f_0(t)$.
- Střední hodnota se nazývá **střední doba do opravy** a je označována zkratkou MTTR (z anglického *mean time to repair*).
- Hazardní funkci se říká **intenzita oprav**.

5.1.1 Intenzita poruch

Typickým tvarem grafu hazardní funkce náhodné veličiny MTTF je tzv. vanová křivka, viz obrázek 5.1. Tento graf se dělí na tři části vymezující tři typická období života komponenty:

I První část křivky je klesající, odpovídající období se nazývá **období časných poruch**. Jedná se o dobu, kdy jsou odhaleny např. výrobní vady.

II Druhá část křivky je konstantní, případně je možný lehký lineární nárůst. V této době dochází k poruchám pouze z vnějších příčin, nazývá se **období stabilního života**.



Obrázek 5.1: Vanová křivka

III V této části křivka roste. Jedná se o **období stárnutí**, k poruchám dochází především z důvodu opotřebení komponenty.

V jednotlivých obdobích bývá intenzita poruch obvykle modelována různými rozděleními pravděpodobnosti, neboť je obtížné vyjádřit celou křivku pomocí jediné funkce. V některých případech také může fáze I nebo III zcela chybět. [2, 3]

5.2 Spolehlivost a pohotovost systému

Definice 5.1 (Spolehlivost) *Spolehlivost $R(t)$ je pravděpodobnost, že až do času t nedojde k poruše.*

Definice 5.2 (Nepohotovost) *Nepohotovost $U(t)$ je pravděpodobnost, že je v čase t systém mimo provoz.*

Definice 5.3 (Pohotovost) *Pohotovost $A(t)$ je pravděpodobnost, že je v čase t systém v provozu. Platí $A(t) = 1 - U(t)$.*

Určit spolehlivost systému $R(t)$ je snadné. Vzhledem k definici spolehlivosti zřejmě platí

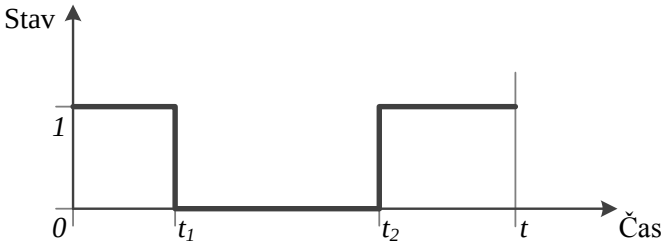
$$R(t) = 1 - F_1(t),$$

kde $F_1(t)$ je distribuční funkce doby do poruchy.

Pro určení pohotovosti $A(t)$ nejprve definujme funkci $A(t \mid k)$ jako pravděpodobnost, že je systém v čase t v provozu, měl-li dosud k poruch. Jev *„systém je v čase t v provozu“* je možné chápat jako souhrn nekonečně mnoha jevů *„systém je v čase t v provozu, měl-li dosud právě k poruch“*, jelikož se tyto jevy navzájem vylučují, pohotovost systému v čase t lze vyjádřit jako následující součet:

$$A(t) = \sum_{k=0}^{\infty} A(t \mid k). \tag{5.1}$$

Pro $k = 0$ se jedná o jev *„až do času t nedošlo k poruše systému“*, pravděpodobnost jeho nastání je z definice rovna spolehlivosti, platí tedy $A(t \mid 0) = R(t)$.



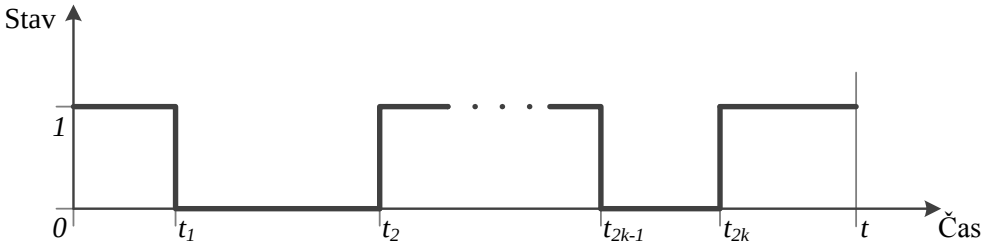
Obrázek 5.2: Elementární jev ω_2

Určeme nyní $A(t \mid 1)$, tedy pravděpodobnost, že je systém v čase t v provozu, došlo-li dosud k právě jedné poruše. Jev *„systém je v čase t v provozu, měl-li dosud právě jednu poruchu“* je možné popsat jako soubor spojitých elementárních jevů ω_2 *„v čase t_1 došlo k poruše, následně v čase t_2 k opravě, od opravy až od času t je systém v provozu“* (viz obrázek 5.2):

1. K selhání došlo v nekonečně malém intervalu dt_1 v čase t_1 s pravděpodobností $f_1(t_1)dt_1$.
2. Selhání bylo následováno opravou v časovém intervalu dt_2 v čase t_2, kde $t_1 < t_2 < t$; čas události opravy se měří od t_1 a pravděpodobnost této události je $f_0(t_2 - t_1)dt_2$.
3. Od času opravy (t_2) do času t setrvává systém v operativním stavu s pravděpodobností $1 - F_1(t - t_2)$.

Tyto tři události jsou nezávislé, pravděpodobnost elementárního jevu ω_2 je proto rovna součinu jejich pravděpodobností, platí $P(\omega_2) = f_1(t_1)\,dt_1 \cdot f_0(t_2 - t_1)\,dt_2 \cdot [1 - F_1(t - t_2)]$. Pravděpodobnost $A(t \mid 1)$ tedy lze vyjádřit následovně:

$$A(t \mid 1) = \int_0^t \int_{t_1}^t f_1(t_1) \cdot f_0(t_2 - t_1) \cdot [1 - F_1(t - t_2)] dt_1 dt_2. \tag{5.2}$$



Obrázek 5.3: Elementární jev ω_{2k}

Událost ω_2 dále zobecníme na k selhání v časech $t_1, t_3, \ldots, t_{2k-1}$ a k oprav v časech $t_2, t_4, \ldots, t_{2k}$. Od času t_{2k} do času t systém setrvává v provozu. Tento jev označme ω_{2k}, viz obrázek 5.3. Pro pravděpodobnost jeho nastání platí

$$P(\omega_{2k}) = f_1(t_1)\,\mathrm{d}t_1 \cdot f_0(t_2 - t_1)\,\mathrm{d}t_2 \cdot \ldots \cdot f_0(t_{2k} - t_{2k-1})\,\mathrm{d}t_{2k}\,[1 - F_1(t - t_{2k})]\,.$$

Pro $A(t \mid k)$ tedy platí

$$A(t \mid k) = \int_0^t \int_{t_1}^t \dots \int_{t_{2k-1}}^t P(\omega_{2k}) = \tag{5.3}$$

$$= \int_0^t \int_{t_1}^t \dots \int_{t_{2k-1}}^t f_1(t_1) f_0(t_2 - t_1) \dots f_0(t_{2k} - t_{2k-1}) [1 - F_1(t - t_{2k})] \, \mathrm{d}t_1 \, \mathrm{d}t_2 \dots \mathrm{d}t_{2k},$$

což můžeme dosadit do (5.1). Pohotovost $A(t)$ je tedy dána nekonečnou sumou vícerozměrných integrálů.

5.2.1 Případ exponenciálního rozdělení

Jak již bylo naznačeno, exponenciální rozdělení se používá pro modelování náhodné veličiny doba do poruchy v období stabilního života. TTF má exponenciální rozdělení s parametrem λ, který je zároveň konstantní hazardní funkcí a je označován jako **intenzita poruch**. Hustota pravděpodobnosti TTF se značí $f_1(t)$, distribuční funkce $F_1(t)$. Platí

$$f_1(t) = \lambda \, \mathrm{e}^{-\lambda t} \text{ a } F_1(t) = 1 - \mathrm{e}^{-\lambda t}.$$

Pro střední hodnotu doby do poruchy platí

$$\mathrm{MTTF} = \int_0^\infty t \cdot \mathrm{e}^{-\lambda t} \, \mathrm{d}t = \lim_{x \to \infty} \int_0^x t \cdot \lambda \, \mathrm{e}^{-\lambda t} \, \mathrm{d}t = \lim_{x \to \infty} \left(-\frac{x}{\mathrm{e}^{\lambda x}} - \frac{1}{\lambda} \cdot \mathrm{e}^{-\lambda x} + \frac{1}{\lambda} \right) = \frac{1}{\lambda}.$$

Podobně TTR má exponenciální rozdělení s parametrem μ, který se nazývá **intenzita oprav**. MTTR je analogicky rovna $\frac{1}{\mu}$ a pro hustotu pravděpodobnosti TTR platí

$$f_0(t) = \mu \, \mathrm{e}^{-\mu t}.$$

Pro pohotovost $A(t)$ v tomto případě platí

$$A(t) = \frac{\mu}{\lambda + \mu} + \frac{\lambda}{\lambda + \mu} \, \mathrm{e}^{-(\lambda + \mu)t}. \tag{5.4}$$

Důkaz.

Po vhodné úpravě integračních mezí a dosazení za $F_1(t - t_{2k})$ přejde (5.3) na tvar

$$A(t \mid k) = \int_0^{t_2} \int_0^{t_3} \dots \int_0^{t_{2k}} \int_0^t f_1(t_1) f_0(t_2 - t_1) \dots f_0(t_{2k} - t_{2k-1}) \, \mathrm{e}^{-\lambda(t - t_{2k})} \, \mathrm{d}t_1 \, \mathrm{d}t_2 \dots \mathrm{d}t_{2k}.$$

Proveďme dále substituci

$$B(k, t_{2k}) = \int_0^{t_2} \int_0^{t_3} \dots \int_0^{t_{2k}} f_1(t_1) f_0(t_2 - t_1) \dots f_0(t_{2k} - t_{2k-1}) \, \mathrm{d}t_1 \, \mathrm{d}t_2 \dots \mathrm{d}t_{2k-1},$$

po dosazení do (5.1) dostáváme

$$A(t) = \int_0^t \sum_{k=0}^{\infty} B(k,t_{2k})\, \mathrm{e}^{-\lambda(t-t_{2k})}\, \mathrm{d}t_{2k} = e^{-\lambda t} + \int_0^t \sum_{k=1}^{\infty} B(k,t_{2k})\, \mathrm{e}^{-\lambda(t-t_{2k})}\, \mathrm{d}t_{2k}. \tag{5.5}$$

Povšimněme si, že $B(k,t_{2k})$ je hustotou pravděpodobnosti času poslední (k-té) opravy. Jelikož v tomto vícerozměrném integrálu nezáleží na pořadí operací, můžeme jej přeuspořádat a popsaný proces chápat jako posloupnost k poruch následovanou posloupností k oprav. Náhodnou veličinu popisující čas, kdy byl systém v provozu, označme t_1^k a její hustotu pravděpodobnosti $g_1\left(t_1^k\right)$. Čas, kdy byl systém mimo provoz, označme t_0^k a příslušnou hustotu pravděpodobnosti $g_0\left(t_0^k\right)$. Hustotu pravděpodobnosti součtu $t_{2k} = t_0^k + t_1^k$ můžeme zapsat jako

$$B(k,t_{2k}) = \int_0^{t_{2k}} g_1(x)\, g_0(t_{2k}-x)\, \mathrm{d}x.$$

Využijme nyní toho, že náhodné veličiny TTF a TTR mají exponenciální rozdělení. Náhodná veličina t_1^k, resp. t_0^k, je v tom případě součtem k náhodných veličin pocházejících z téhož exponenciálního rozdělení a má tedy Erlangovo rozdělení s parametry k a λ, resp. k a μ. Podle (2.1) tedy platí

$$\begin{aligned} B(k,t_{2k}) &= \int_0^{t_{2k}} \frac{\lambda\left(\lambda t_{2k}\right)^{k-1} e^{-\lambda t_{2k}}}{(k-1)!} \cdot \frac{\mu\left(\mu\left(t_{2k}-x\right)\right)^{k-1} e^{-\mu(t_{2k}-x)}}{(k-1)!}\, \mathrm{d}x = \\ &= \frac{\lambda^k \mu^k}{(k-1)!\,(k-1)!} \int_0^{t_{2k}} t_{2k}^{k-1} e^{-\lambda t_{2k}} \left(t_{2k}-x\right)^{k-1} e^{-\mu(t_{2k}-x)}\, \mathrm{d}x. \end{aligned}$$

Pro další postup je potřebná znalost Laplaceovy transformace, potřebnou teorii lze najít v [12]. Platí

$$\mathscr{L}\left(B(k,t_{2k})\right) = \left[\frac{\lambda}{s+\lambda} \cdot \frac{\mu}{s+\mu}\right]^k.$$

Dále určíme Laplaceovu transformaci sumy $\sum_{k=1}^{\infty} B(k,t_{2k})$

$$\mathscr{L}\left(\sum_{k=1}^{\infty} B(k,t_{2k})\right) = \sum_{k=1}^{\infty} \left[\frac{\lambda}{s+\lambda} \cdot \frac{\mu}{s+\mu}\right]^k = \frac{\lambda\mu}{s(s+\lambda+\mu)}$$

jako součet geometrické řady a provedeme zpětnou transformaci:

$$\sum_{k=1}^{\infty} B(k,t_{2k}) = \frac{\lambda\mu\left(1-e^{-(\lambda+\mu)t_{2k}}\right)}{\lambda+\mu}.$$

Po dosazení do (5.5) a následné integraci získáváme právě vzorec (5.4) pro výpočet pohotovosti systému o jedné komponentě v čase t. [1] ∎

6 Výpočet pohotovosti systému

Tato kapitola je stěžejní částí celé bakalářské práce, jejím úkolem je popsat algoritmy pro výpočet pohotovosti systému, jehož komponenty se navzájem neovlivňují. Náhodné veličiny doba do poruchy a doba do opravy každé ze systémových komponent jsou modelovány pomocí exponenciálního rozdělení.

Pohotovost systému je kvantifikována nejprve analyticky, poté je na tento problém aplikována metoda Monte Carlo. Nedílnou součástí kapitoly je také srovnání těchto dvou metod a zhodnocení, kdy je vhodnější použít metodu Monte Carlo a kdy naopak analytické řešení.

6.1 Zadání úlohy a základní poznatky

Jak již bylo naznačeno, je řešena obecná úloha s tímto zadáním:

> *Je dán systém s n nezávislými komponentami a příslušná systémová funkce $S(\mathbf{B})$. Dále je známa intenzita poruch λ_i a intenzita oprav μ_i každé z komponent. Určete pohotovost systému v čase t.*

Je třeba shrnout některé poznatky, jichž bude využíváno v obou metodách výpočtu.

Stavový prostor V zadaného systému čítá 2^n vektorů. Dosazením jednotlivých stavových vektorů do systémové funkce lze stavový prostor rozdělit na dvě disjunktní podmnožiny:

$$\begin{aligned} V_0 &= \{\mathbf{B} \in V \mid S(\mathbf{B}) = 0\}, \\ V_1 &= V \setminus V_0. \end{aligned}$$

Je-li určována pohotovost systému v čase t, je vlastně vyčíslována pravděpodobnost, že je v tomto čase systém v provozu.

Pro určení pravděpodobnosti $p_i(t)$, že je v čase t v provozu i-tá komponenta, lze použít vzorec

$$p_i(t) = \frac{\mu_i}{\lambda_i + \mu_i} + \frac{\lambda_i}{\lambda_i + \mu_i} e^{-(\lambda_i + \mu_i)t} \tag{6.1}$$

odvozený v sekci 5.2.1.

Dále je třeba vyjádřit pravděpodobnost $P(\mathbf{B},t)$, že se v čase t systém nachází ve stavu odpovídajícím konkrétnímu stavovému vektoru $\mathbf{B} = (b_1, b_2, \ldots, b_n)$. Označíme-li

$$\beta_i(t) = \begin{cases} p_i(t) & \text{jestliže } b_i = 1 \\ 1 - p_i(t) & \text{jestliže } b_i = 0, \end{cases}$$

lze hledanou pravděpodobnost $P(\mathbf{B},t)$ zapsat vztahem

$$P(\mathbf{B},t) = \prod_{i=1}^{n} \beta_i(t). \tag{6.2}$$

Nyní již lze přistoupit ke konkrétním metodám řešení. V obou případech bude hledána nepohotovost $U(t)$, pohotovost pak bude vyjádřena jako $1 - U(t)$. [1]

6.2 Kvantifikace pohotovosti systému s nezávislými prvky analyticky

Nepohotovost systému $U(t)$ v čase t je pravděpodobnost, že je systém v tomto čase mimo provoz, tedy že se nachází ve stavu patřícím do množiny V_0. Tato pravděpodobnost je součtem pravděpodobností $P(\mathbf{B}_j,t)$, že je v tomto čase systém ve stavu $\mathbf{B}_j \in V_0$. Nepohotovost tedy lze vyjádřit následovně:

$$U(t) = P(S(\mathbf{B}) = 0,t) = \sum_{j=1,\mathbf{B}_j \in V_0}^{2^n} P(\mathbf{B}_j,t) = \sum_{j=1,\mathbf{B}_j \in V_0}^{2^n} \left(\prod_{i=1}^{n} \beta_i^j(t) \right). \tag{6.3}$$

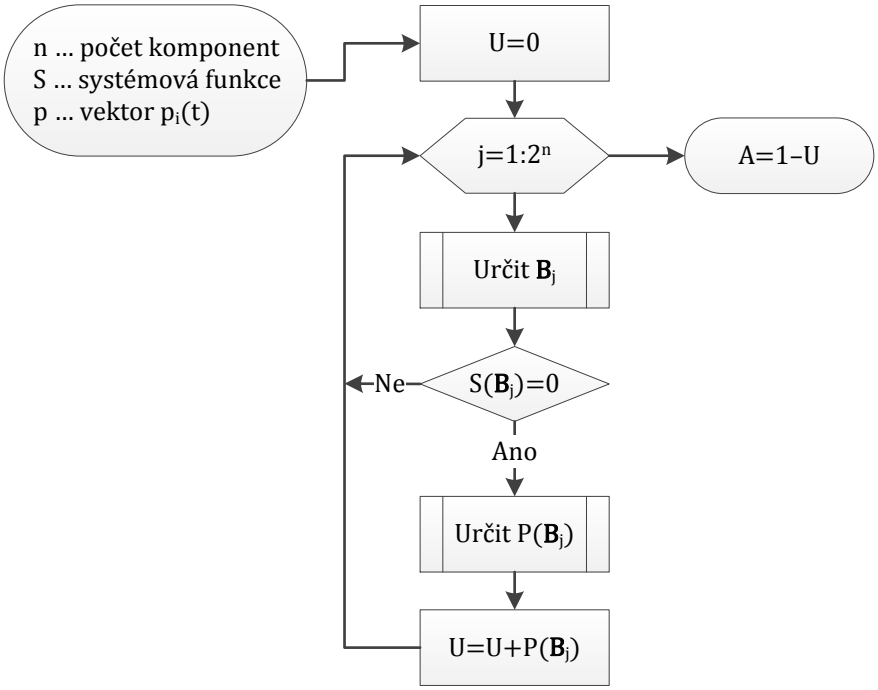
Rovnost (6.3) vede na jednoduchý algoritmus pro kvantifikaci nepohotovosti systému v určitém čase t. Pomocí vypočtené nepohotovosti $U(t)$ již snadno vyjádříme pohotovost systému $A(t)$ v čase t, platí $A(t) = 1 - U(t)$.

Algoritmus 6.1 (Analytický výpočet pohotovosti systému)

1. Inicializovat proměnnou U, $U = 0$.
2. Projít všech 2^n stavových vektorů ze stavového prostoru V.
   (a) Pro každý stavový vektor $\mathbf{B}_j$ ověřit, zda platí $S(\mathbf{B}_j) = 0$.
      i. Pokud ano, vypočítat pravděpodobnost $P(\mathbf{B}_j,t)$ tohoto stavu podle (6.2) a přičíst ji k nepohotovosti U.
   (b) Pokračovat dalším vektorem, tedy vrátit se k bodu (a).
3. Určit pohotovost systému v uvažovaném čase jako $1 - U$.

```
function [ res ] = Pohotovost_analyticky( S, n, p )
%   S   systémová funkce (handle)
%   n   počet komponent systému
%   p   vektor pravděpodobností, že je i-tá komponenta
%       v uvažovaném čase v provozu
U=0;
for j=1:2^n
    B=r2B(j,n);          % funkce pro určení j-tého stavového vektoru
    if S(B)==0           % dosazení stavového vektoru do systémové funkce
        U=U+Pr(B,n,p);   % inkrementace nedostupnosti U
    end
end
res=1-U;                 % hledaná pohotovost
```

Výpis 6.1: Zdrojový kód algoritmu pro analytický výpočet pohotovosti systému



Obrázek 6.1: Vývojový diagram algoritmu pro analytický výpočet pohotovosti systému

6.3 Řešení pomocí metody Monte Carlo

Při hledání nepohotovosti $U(t)$ metodou Monte Carlo lze postupovat podle kroků uvedených v sekci 3.2.

Nejprve je třeba určit náhodnou veličinu X vhodnou k simulování této úlohy. Touto náhodnou veličinou je ohodnocení stavového vektoru $\mathbf{B}_j$ ze stavového prostoru V podle toho, zda je pro tento stavový vektor systém v provozu či nikoliv, tedy

$$X = \begin{cases} 1 & \text{když } S(\mathbf{B}_j) = 0 \\ 0 & \text{když } S(\mathbf{B}_j) \neq 0. \end{cases} \tag{6.4}$$

Je třeba ověřit, že NV X je skutečně vhodným odhadem nepohotovosti systému, tedy ukázat, že střední hodnota X je rovna nepohotovosti $U(t)$. Platí

$$E(X) = \sum_{j=1}^{2^n} X_j \cdot P(\mathbf{B}_j, t) = \sum_{j=1, \mathbf{B}_j \in V_0}^{2^n} P(\mathbf{B}_j, t) = U(t),$$

viz (6.3), náhodná veličina X je tedy zvolena vhodně.

Náhodný pokus spočívá v generování náhodných stavových vektorů tak, aby platilo, že konkrétní stavový vektor $\mathbf{B}_j$ bude vygenerován s pravděpodobností $P(\mathbf{B}_j, t)$, a v následném určení hodnoty X_j podle (6.4). Pravděpodobnost $p_i(t)$, že je i-tá komponenta v čase t v provozu, je známa, resp. lze vypočíst pomocí (6.1). Náhodný stavový vektor $\mathbf{B}_j$ lze tedy určit pomocí tohoto algoritmu:

Algoritmus 6.2 (Náhodný výběr stavového vektoru)

1. Inicializovat nulový vektor $\mathbf{B}_j$ o délce n.
2. Projít všech n komponent.
   (a) Pro i-tou komponentu vygenerovat náhodné číslo ξ z $R(0;1)$.
   (b) Je-li $\xi < p_i(t)$, přepsat hodnotu na i-té pozici vektoru $\mathbf{B}_j$ na 1.

Po N opakováních tohoto pokusu je získán náhodný výběr $X_1, X_2, \ldots X_N$. Výběrový průměr $\bar{X}$ tohoto náhodného výběru je odhadem střední hodnoty náhodné veličiny X, a tedy i nepohotovosti $U(t)$. Pohotovost systému lze tedy odhadnout hodnotou $1 - \bar{X}$.

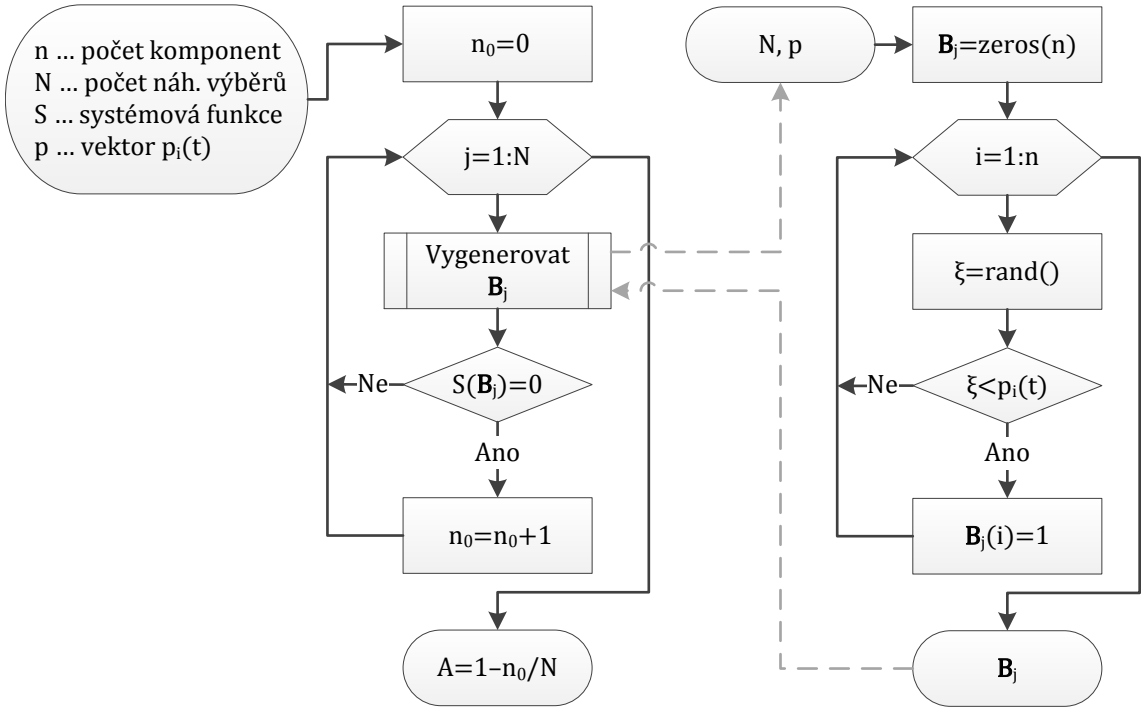
Postup výpočtu pohotovosti systému shrnuje následující algoritmus:

Algoritmus 6.3 (Výpočet pohotovosti systému metodou Monte Carlo)

1. Inicializovat proměnnou n_0 na hodnotu 0. Tato proměnná reprezentuje počet vygenerovaných náhodných vektorů, pro které je systém mimo provoz.
2. Vygenerovat N náhodných stavových vektorů $\mathbf{B}_j$ pomocí algoritmu 6.2.
   (a) Pro každý $\mathbf{B}_j$ určit, zda platí $S(\mathbf{B}_j) = 0$.
      i. Pokud ano, inkrementovat n_0 o hodnotu 1.
   (b) Pokračovat dalším vektorem, tedy bodem (a).
3. Určit pohotovost systému v uvažovaném čase jako $1 - \frac{n_0}{N}$.
4. Kvantifikovat přesnost řešení.

```
function [ res, epsilon, PRSD ] = Pohotovost_MC( S, n, p, N, alpha )
%   S   systémová funkce (handle)
%   n   počet komponent systému
%   p   vektor pravděpodobností, že je i-tá komponenta
%       v uvažovaném čase v provozu
%   N   počet náhodných výběrů
n_0=0;  % počet stavových vektorů, pro které je systém mimo provoz
for i=1:N
    B=rand(1,n)<p;      % vygenerován náhodný stavový vektor
    if S(B)==0          % dosazení stavového vektoru do systémové funkce
        n_0=n_0+1;
    end
end
res=1-n_0/N;                        % hledaná pohotovost
s=sqrt((N-n_0-n_0*n_0/N)/(N-1));    % výběrová směrodatná obchylka
epsilon=s*norminv(1-alpha/2)/sqrt(N);
PRSD=100*s/((n_0/N)*sqrt(N));
```

Výpis 6.2: Zdrojový kód algoritmu pro výpočet pohotovosti systému metodou MC



Obrázek 6.2: Vývojový diagram algoritmu pro výpočet pohotovosti metodou MC

6.3.1 Přesnost řešení

Pro určení *PRSD* a odchylky ε od přesného řešení (s hladinou významnosti α) je třeba znát výběrovou směrodatnou odchylku s. Díky tomu, že má náhodná veličina X alternativní rozdělení, stačí pro určení s znát pouze počet pokusů, ve kterých NV nabyla hodnoty 0, tedy hodnotu n_0. Hodnoty 1 tedy nabyla v $N - n_0$ případech. Platí

$$s = \sqrt{\frac{1}{N-1} \cdot \left(\sum_{i=1}^{N} X_i^2 - N\bar{X}^2 \right)} = \sqrt{\frac{1}{N-1} \cdot \left((N-n_0) - \frac{n_0^2}{N} \right)}.$$

Tuto hodnotu již stačí dosadit do vzorců pro výpočet ε a *PRSD*.

6.4 Modifikace úlohy

6.4.1 Výpočet výkonu systému

Oborem hodnot systémové funkce nemusí být vždy pouze množina $\{0,1\}$. Není-li chápána jako logická funkce, může po dosazení stavových indikátorů jednotlivých komponent vypovídat o jakékoliv vlastnosti systému, například systémová funkce z příkladu 4.1 vyjadřuje průtok potrubím v uvažovaném čase. Jiným měřítkem výkonu systému může být rychlost výroby na montážní lince, výkon generátoru elektrické energie, apod. Výkonem tedy dále není myšlen pouze elektrický výkon (vyprodukovaná elektrická energie za jednotku času), ale rychlost výroby libovolného produktu.

Je-li zadána systémová funkce tohoto typu, je možné určit pravděpodobný výkon $Q(t)$ (případně rychlost produkce) systému v určitém čase t analyticky pomocí vzorce

$$Q(t) = \sum_{j=1}^{2^n} P(\mathbf{B}_j, t) \cdot S(\mathbf{B}_j), \tag{6.5}$$

který lze jednoduše algoritmizovat:

Analytický výpočet výkonu systému

1. Inicializovat proměnnou Q, $Q = 0$.
2. Projít všech 2^n stavových vektorů ze stavového prostoru V.
   (a) Pro každý stavový vektor $\mathbf{B}_j$ inkrementovat Q o hodnotu $S(\mathbf{B}_j) \cdot P(\mathbf{B}_j, t)$.
3. Pravděpodobný výkon systému v uvažovaném čase je roven Q.

Pravděpodobný výkon systému v čase t lze rovněž spočíst pomocí metody MC. Náhodnou veličinou X je zde hodnota systémové funkce pro náhodně vygenerovaný stavový vektor $\mathbf{B}_j$. Střední hodnota této NV je rovna pravděpodobnému výkonu systému v uvažovaném čase. Algoritmus se zásadně neliší od toho pro výpočet pohotovosti systému, rozdíl však je v určování přesnosti. NV X již nemá alternativní rozdělení, může nabýt až 2^n hodnot, je tedy nutné průběžné počítat sumu X_i^2 pro následné dosazení do vzorce pro výpočet výběrové směrodatné odchylky s.

Algoritmus 6.4 (Výpočet výkonu systému metodou Monte Carlo)

1. Inicializovat proměnné q a m na hodnotu 0.
2. Vygenerovat N náhodných stavových vektorů $\mathbf{B}_j$ pomocí algoritmu 6.2.
   (a) Pro každý $\mathbf{B}_j$ inkrementovat q o hodnotu $S(\mathbf{B}_j)$.
   (b) Pro každý $\mathbf{B}_j$ inkrementovat m o hodnotu $(S(\mathbf{B}_j))^2$.
3. Určit pravděpodobný výkon systému v uvažovaném čase jako $\frac{q}{N}$.
4. Určit výběrovou směrodatnou odchylku s jako $\sqrt{\frac{1}{N-1} \cdot \left(m - \frac{q^2}{N}\right)}$, určit ε a *PRSD*.

6.4.2 Výpočet průměrného výkonu v intervalu

Pro určení pravděpodobného výkonu $Q(t)$ systému v čase t použijeme vzorec (6.5). Průměrný výkon $Q_p(t_0;t_1)$ v intervalu $(t_0;t_1)$ pak vyjádříme jako

$$\begin{aligned} Q_p(t_0;t_1) &= \frac{1}{t_1-t_0}\int_{t_0}^{t_1} Q(t)\,\mathrm{d}t = \frac{1}{t_1-t_0}\int_{t_0}^{t_1}\sum_{j=1}^{2^n} P(\mathbf{B}_j,t)\cdot S(\mathbf{B}_j)\,\mathrm{d}t = \\ &= \frac{1}{t_1-t_0}\sum_{j=1}^{2^n} S(\mathbf{B}_j)\cdot\int_{t_0}^{t_1} P(\mathbf{B}_j,t)\,\mathrm{d}t. \end{aligned} \tag{6.6}$$

Pomocí tohoto vzorce lze průměrný výkon systému vypočítat analyticky. Je však nutné provést celkem 2^n integrací, což může být náročné již pro systémy o nízkém počtu komponent, viz sekce 8.2.1.

Efektivněji lze úlohu řešit metodou MC. Náhodnou veličinou je zde hodnota systémové funkce v náhodně vygenerovaném čase z intervalu $(t_0;t_1)$. Střední hodnota této NV udává právě pravděpodobnou průměrnou hodnotu systémové funkce v intervalu $(t_0;t_1)$.

Algoritmus 6.5 (Výpočet průměrného výkonu metodou Monte Carlo)

1. Inicializovat proměnné q a m na hodnotu 0.
2. Vygenerovat N náhodných čísel ξ_j z intervalu $(t_0;t_1)$ pomocí přepočtu z $R(0;1)$. Pro každý čas ξ_j:
   (a) Vygenerovat náhodný stavový vektor $\mathbf{B}_j$ pomocí algoritmu 6.2.
   (b) Inkrementovat q o hodnotu $S(\mathbf{B}_j)$ a m o hodnotu $(S(\mathbf{B}_j))^2$.
3. Určit průměrný výkon systému jako $\frac{q}{N}$, určit ε a *PRSD* stejně jako v algoritmu 6.4.

Vynásobíme-li $Q_p(t_0;t_1)$ délkou časového intervalu, tedy hodnotou (t_1-t_0), získáme množství produktu, které systém pravděpodobně vyrobí od času t_0 do času t_1.

Vrací-li systémová funkce pouze hodnoty 0 a 1 podle toho, jestli je systém v provozu, je výsledkem algoritmu průměrná pohotovost v intervalu $(t_0;t_1)$. Vynásobíme-li tuto hodnotu číslem (t_1-t_0), získáme dobu, po kterou bude systém pravděpodobně v provozu během intervalu $(t_0;t_1)$.

Zdrojové kódy algoritmů 6.4.1, 6.4 a 6.5 jsou uvedeny v příloze A.1.

6.4.3 Alternativy systémové funkce

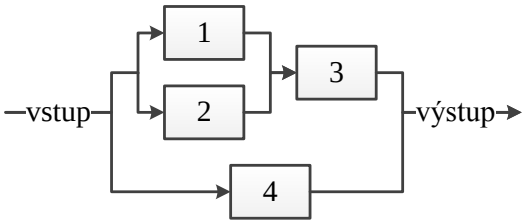
Je-li určována pohotovost systému, je třeba znát mechanismus, který umožní určit, zda je pro konkrétní stavový vektor $\mathbf{B} = (b_1, b_2, \ldots, b_n)$ systém v provozu, či nikoliv. Takovým mechanismem nemusí nutně být systémová funkce. Místo ní může být zadána například pravdivostní tabulka nebo matice sousednosti reprezentující strukturu systému.

Pravdivostní tabulka Jedná se o tabulku o 2^n řádcích a $n+1$ sloupcích, kde n je počet komponent systému. Prvních n sloupců reprezentuje všech 2^n stavů systému. Poslední sloupec obsahuje nuly a jedničky podle toho, zda je pro daný stavový vektor systém v provozu. Jako vstup pochopitelně nemusí být zadávána celá tabulka, postačí poslední sloupec, tedy sloupcový vektor o délce 2^n.

Určit, zda je systém pro daný stav komponent v provozu, je snadné, stačí přečíst hodnotu v příslušném řádku tohoto sloupcového vektoru.

Matice sousednosti Maticí sousednosti je čtvercová matice o rozměrech $(n+2)\times(n+2)$, kde n je počet komponent systému, jehož strukturu tato matice reprezentuje. První řádek a sloupec matice je rezervován pro vstupní systémový prvek, poslední řádek a sloupec pak pro výstup. Ostatní řádky a sloupce reprezentují systémové komponenty. Matice je tvořena pouze čísly z množiny $\{0,1\}$, resp. $\{\text{false, true}\}$. Číslo 1 na pozici $[a,b]$ značí, že existuje přímá orientovaná cesta z prvku a do prvku b, viz příklad 6.1.

Pro rozhodnutí o průchodnosti systému (tj. pro určení, zda je pro daný stavový vektor $\mathbf{B}$ systém zadaný maticí sousednosti v provozu) jsou použity funkce uvedené v příloze A.2.



Obrázek 6.3: Schéma systému z příkladu 6.1

Příklad 6.1
Strukturu jistého systému zachycuje schéma na obrázku 6.3. Tuto strukturu je možné zapsat maticí sousednosti

$$\begin{array}{c} \\ \rightarrow \\ 1 \\ 2 \\ 3 \\ 4 \\ \leftarrow \end{array} \begin{array}{c} \begin{array}{cccccc} \downarrow & 1 & 2 & 3 & 4 & \uparrow \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

nebo také pomocí pravdivostní tabulky 6.1, respektive jako vektor posledního sloupce

$$v = (0,1,0,1,0,1,1,1,0,1,1,1,0,1,1,1)^T.$$

△

b_1	b_2	b_3	b_4	$S(\mathbf{B})$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1

b_1	b_2	b_3	b_4	$S(\mathbf{B})$
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Tabulka 6.1: Tabulková forma systémové funkce z příkladu 6.1

6.5 Srovnání metod

Porovnávat obě metody z hlediska přesnosti nemá smysl, v případě analytického řešení je získán přesný výsledek, zatímco v případě metody Monte Carlo pouze jeho odhad. Metoda MC tedy nemůže být přesnější.

Vhodným kritériem pro porovnání metod je výpočetní čas. Je třeba zjistit, pro jak rozsáhlé systémy je možné výpočet realizovat. Pro testování obou způsobů výpočtu je zvolena tato úloha:

> *Je dán systém o n komponentách, jež jsou řazeny sériově. Pravděpodobnost p, že je i-tá komponenta v jistém čase v provozu je rovna* 0.999. *Určete pohotovost systému v čase t pomocí algoritmů 6.1 (analyticky) a 6.3 (metodou MC).*

Pohotovost takto jednoduchého systému by pochopitelně nebylo nutné řešit těmito algoritmy, nicméně pro porovnání metod je systém vhodný. Využijeme právě toho, že lze pohotovost v uvažovaném čase t vypočíst jednoduše pomocí vzorce

$$A(t) = p^n, \tag{6.7}$$

kde n je počet komponent. Je tedy možné porovnat výsledky metody MC s přesným řešením i v případě systémů o velkém počtu komponent, které již z důvodu časové náročnosti nejsou pomocí algoritmu 6.1 řešitelné.

Úloha byla řešena pro různá n analyticky i metodou MC za stejných podmínek na notebooku s procesorem Intel Core i7. Systém byl zadán nejprve maticí sousednosti ve tvaru

$$\begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & & 0 & 0 \\ 0 & 0 & 0 & & 0 & 0 \\ \vdots & & & \ddots & & \\ 0 & 0 & 0 & & 0 & 1 \\ 0 & 0 & 0 & & 0 & 0 \end{pmatrix}$$

a poté systémovou funkcí

$$S(\mathbf{B}) = b_1 \cdot b_2 \cdot \ldots \cdot b_n,$$

kde n je počet komponent. Byl zaznamenáván především výpočetní čas, v případě zadání systému systémovou funkcí značen τ_F, v případě matice sousednosti τ_M.

V případě metody MC byl zaznamenáván také odhad nepohotovosti systému $\bar{X}$, skutečná chyba řešení $|1-\bar{X}-A(t)|$ a přesnost vyjádřená pomocí *PRSD* a ε při hladině významnosti $\alpha = 0.05$. Výběrový soubor měl rozsah 10^5. Výsledky pro některá n jsou uvedeny v tabulkách 6.2 (systém zadáván systémovou funkcí) a 6.3 (systém zadáván maticí sousednosti), rozsáhlejší tabulky lze nalézt v příloze (B.2 a B.3). Z těchto tabulek je patrné, že výpočetní čas roste přibližně lineárně vzhledem k počtu komponent.

n	$\bar{X}$	$1-\bar{X}$	$A(t)$ podle (6.7)	$\lvert 1-\bar{X}-A(t)\rvert$	ε ($\alpha = 0.05$)	$PRSD$	τ_F [s]
10	0,00987	0,99013	0,99004	0,00009	0,00061	3,17%	0,6537
20	0,01998	0,98002	0,98019	0,00017	0,00087	2,21%	0,7181
50	0,04955	0,95045	0,95121	0,00076	0,00135	1,38%	0,8872
100	0,09523	0,90477	0,90479	0,00002	0,00182	0,97%	0,9702
200	0,18109	0,81891	0,81865	0,00026	0,00239	0,67%	1,2583
400	0,33026	0,66974	0,67019	0,00045	0,00291	0,45%	1,996

Tabulka 6.2: Řešení metodou MC, zadána systémová funkce

n	$\bar{X}$	$1-\bar{X}$	$A(t)$ podle (6.7)	$\lvert 1-\bar{X}-A(t)\rvert$	ε ($\alpha = 0.05$)	$PRSD$	τ_M [s]
10	0,00968	0,99032	0,99004	0,00028	0,00061	3,20%	7,288
20	0,02001	0,97999	0,98019	0,0002	0,00087	2,21%	11,84
50	0,04875	0,95125	0,95121	0,00004	0,00133	1,40%	26,66
100	0,09614	0,90386	0,90479	0,00093	0,00183	0,97%	53,76
200	0,18224	0,81776	0,81865	0,00089	0,00239	0,67%	120,5
400	0,32842	0,67158	0,67019	0,00139	0,00291	0,45%	287,8

Tabulka 6.3: Řešení metodou MC, zadána matice sousednosti

Výsledky analytického řešení pro některá n znázorňuje tabulka 6.4. Je patrné, že s rostoucím n roste výpočetní čas exponenciálně. Po zvýšení počtu komponent o 1, se výpočetní čas přibližně zdvojnásobí, což odpovídá faktu, že je vyhodnocováno všech 2^n stavů systému. Je-li systém zadán maticí sousednosti, výpočet trvá déle než v případě systémové funkce, neboť pro každý stavový vektor je nutné vyhodnocovat průchodnost systému. Kompletní tabulka pro $n \in \{1,2,\ldots,24\}$ je uvedena v příloze (B.1).

n	2^n	$A(t)$	$U(t)$	τ_F [s]	τ_M [s]
5	32	0,99500999	0,00499001	0,000563438	0,001442991
10	1024	0,99004488	0,00995512	0,011270815	0,040868155
15	32768	0,985104546	0,014895454	0,359183579	1,451981013
20	1048576	0,980188865	0,019811135	12,87711559	47,8219628
21	2097152	0,979208676	0,020791324	24,43293188	95,72386145
22	4194304	0,978229467	0,021770533	48,93312174	193,9155361
23	8388608	0,977251238	0,022748762	99,80701629	392,5519846
24	16777216	0,976273987	0,023726013	201,1574471	780,3699182

Tabulka 6.4: Analytické řešení

Podle tabulky 6.4 se dá odhadovat, že vypočíst analyticky pohotovost systému o 30 komponentách by trvalo přes 3 hodiny, o 40 komponentách přibližně 140 dní a o 50 komponentách více než 400 let. Výpočet je sice možné zefektivnit například paralelizací, i tak jej však pro vysoké počty komponent nebude možné realizovat. Přitom metodou Monte Carlo trval výpočet pohotovosti systému o 50 komponentách s uvedenou přesností necelou sekundu.

Je třeba mít na paměti, že uvažovaný systém měl velice specifickou strukturu, v případě obecného systému se může lišit zejména čas nutný k vyhodnocení průchodnosti systému zadaného maticí sousednosti. Pohotovost obecnějších systému je řešena v kapitole 8.

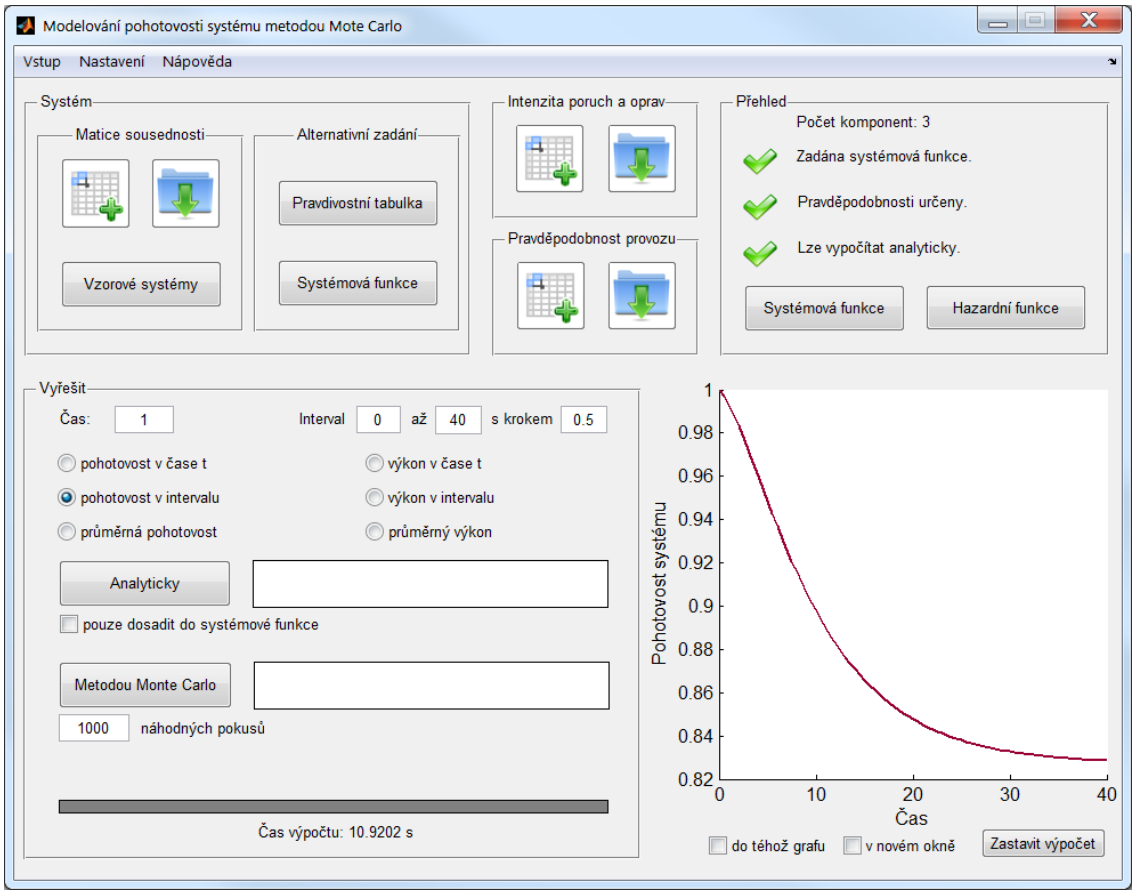
7 Implementace programu a paralelizace

Algoritmy uvedené v předchozí kapitole byly implementovány v podobě funkcí v jazyce Matlab. Následně byl vytvořen program s grafickým uživatelským rozhraním (viz obrázek 7.1), který zahrnuje všechny tyto funkce. Umožňuje tak řešení pohotovosti systémů zadaných různými způsoby a souvisejících úloh metodou Monte Carlo i analyticky.

Dále byly vytvořeny funkce, které pro určení pohotovosti systému zadaného maticí sousednosti využívají paralelní výpočty.

7.1 Popis vytvořeného programu

Předmětem této sekce není popis použitých kódů, neboť ty nejdůležitější již byly uvedeny a další lze najít v textové příloze a na přiloženém CD, zabývá se spíše uživatelským popisem funkčnosti programu. Více informací je uvedeno v nápovědě k programu, která je spustitelná přímo z GUI.



Obrázek 7.1: Grafické uživatelské rozhraní

7.1.1 Typy úloh

Úlohy řešitelné tímto programem jsou rozděleny na 6 typů:

1. Určení pohotovosti systému v zadaném čase.
2. Modelování pohotovosti systému v průběhu zadaného časovém intervalu.
3. Výpočet průměrné pohotovost systému v daném časovém intervalu.
4. Výpočet pravděpodobného výkonu systému v zadaném čase.
5. Modelování pravděpodobného výkonu systému v časovém intervalu.
6. Určení průměrného výkonu systému v časovém intervalu.

Typy 3. a 6. jsou řešitelné pouze metodou Monte Carlo, ostatní lze řešit i analyticky.

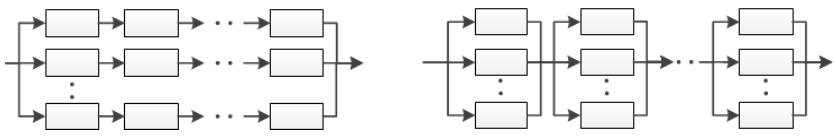
7.1.2 Práce s programem

Zadání systému

Systém je možné zadat pomocí matice sousednosti, systémové funkce, nebo případně pomocí pravdivostní tabulky. Matici sousednosti a pravdivostní tabulku lze zadat přímo do GUI nebo načíst ze souboru. Systémová funkce je zadávána formou textového řetězce. Při řešení posledních tří typů úloh je systém zadáván pomocí systémové funkce vyjadřující výkon systému.

Při testování bylo zjištěno, že nejefektivnějším způsobem zadání systému je obvykle systémová funkce. Nicméně snadnější než vytvořit systémovou funkci obvykle bývá sestavit matici sousednosti. Proto byl implementován také algoritmus pro převod matice sousednosti na odpovídající systémovou funkci.

Je-li systém zadán maticí sousednosti, je možné vytvořit tzv. „bloky *k*/*m*“, neboli skupiny *m* komponent, které jsou v provozu právě tehdy, když je v provozu alespoň *k* z nich. (Bude dále vysvětleno v sekci 8.4.)



Obrázek 7.2: Vzorové sério-paralelní kombinace

Součástí programu je možnost vygenerování systémové funkce a matice sousednosti systémů, jejichž struktura odpovídá některému ze schémat na obrázku 7.2. Mezi vzorovými systémy jsou dále zařazeny příklady řešené v následující kapitole.

Zadání hazardních funkcí

Intenzita poruch a oprav je zadávána maticí o rozměrech $2 \times n$, kde n je počet komponent systému. Na prvním řádku je intenzita poruch a na druhém intenzita oprav jednotlivých komponent. Také je možné zadat přímo pravděpodobnosti, že jsou systémové komponenty v provozu v konkrétním čase, v němž má být určována pohotovost či výkon systému.

Výstup

Výstupem je vždy hledaná hodnota pohotovosti či výkonu systému, nebo vhodné grafické znázornění výsledku. V případě úloh typu 2. (resp. 5.) je vykreslena lomená čára znázorňující závislost pohotovosti systému (resp. pravděpodobného výkonu) na čase v zadaném intervalu, s daným krokem.

V případě řešení metodou MC je vždy určena (a případně graficky znázorněna) také přesnost řešení. Výchozí hodnota hladiny významnosti je 0.05, je však možné ji změnit v Nastavení.

7.2 Paralelizace

Tato sekce je věnována paralelizaci základních algoritmů pro výpočet pohotovosti systému, jež je zadán maticí sousednosti, metodou MC i analyticky. Úloha bude rozdělena do několika současně probíhajících vláken, z nichž každé vykoná část výpočtu. Cílem je optimalizace algoritmů a dosažení kratšího výpočetního času.

Metoda MC je vhodná k paralelizaci, neboť jednotlivé náhodné pokusy jsou na sobě nezávislé a mohou být řešeny v různých vláknech. Je-li určována pohotovost systému zadaného maticí sousednosti, jsou v jednotlivých vláknech generovány náhodné stavové vektory a poté je určováno, zda je pro vygenerovaný stavový vektor systém v provozu. Pro vypočtení pohotovosti stačí určit počet vektorů, pro které je systém v provozu a vydělit jej celkovým počtem vygenerovaných vektorů.

Podobně v případě analytického řešení lze systémové vektory, pro něž je třeba vyhodnotit průchodnost, rozdělit na několik částí a ty vyhodnocovat v různých vláknech.

7.2.1 Paralelní cyklus v Matlabu

Matlab umožňuje jednoduchou paralelizaci cyklů s pevným počtem opakování pomocí klíčového slova `parfor`. Počet vláken, v nichž výpočet probíhá, je roven nejvýše počtu jader procesoru. Zdrojové kódy funkcí pro paralelní výpočet pohotovosti systému ve čtyřech vláknech jsou uvedeny v příloze A.3.1.

7.2.2 Technologie CUDA

Efektivněji lze tyto algoritmy paralelizovat pomocí technologie CUDA (Compute Unified Device Architecture). Tato technologie umožňuje využívat grafické karty značky NVIDIA k paralelním výpočtům.

Jednotlivá vlákna jsou zde organizována do bloků, tyto bloky pak do mřížky. V případě grafické karty použité v této simulaci může v každém bloku být maximálně 1024 vláken, bloků v mřížce pak je maximálně 65535×65535. Dále nesmí být překročeno maximální množství využitelné paměti.

V jednotlivých vláknech jsou spouštěny tzv. CUDA kernely, tyto speciální funkce se zapisují do souborů s příponou *.cu*, syntaxe je podobná jazyku C++. Kódy CUDA kernelů pro určení průchodnosti systému metodou MC i analyticky jsou uvedeny v příloze A.3.3. Pří jejich tvorbě bylo čerpáno z [10].

Bitové operace

Kratšího výpočetního času a nízkých paměťových nároků je dosaženo také díky používání tzv. bitových operací. Jsou užívány proměnné a vektory proměnných datového typu `unsigned int` o délce 32 bitů. Binární zápis takové proměnné se skládá ze 32 číslic $(0/1)$, z nichž každá reprezentuje stav jednoho systémového prvku. Při analytickém výpočtu je pro zapsání jednoho stavového vektoru využívána pouze jedna proměnná délky 32 bitů, výpočet metodou MC je optimalizován i pro delší stavové vektory, tj. pro systémy o vyšším počtu komponent.

Práce s kernely v Matlabu

Pracovat se soubory s příponou *.cu*, které obsahují CUDA kernely, je možné přímo z prostředí Matlab. Nejprve je však potřeba pomocí externího kompilátoru (NVIDIA CUDA Compiler) vytvořit také soubor PTX (z anglického *Parallel Thread Execution*). Z těchto dvou souborů lze v Matlabu vytvořit objekt kernel.

Matlab dále umožňuje alokaci polí na grafické kartě, tato pole jsou používána jako vstupní i výstupní proměnné kernelů. Pro realizaci náhodných pokusů v metodě MC, zde se jedná o generování náhodných stavových vektorů, je na grafické kartě předem vytvořeno pole náhodných čísel z $R(0;1)$ o délce $n \cdot N$, kde n je počet komponent systému a N je počet prováděných náhodných pokusů. Pro vygenerování i-tého stavového vektoru je tak použito n čísel z tohoto pole od pozice $i \cdot n$ dále (při indexování od 0).

Zdrojový kód funkcí pracujících s CUDA kernely z prostředí Matlab je uveden v příloze A.3.2.

7.2.3 Porovnání výpočetního času

Pro testování je zvolena stejná úloha jako v sekci 6.5, jde o určení pohotovosti systému, jehož komponenty jsou řazeny sériově. Je tak možné porovnat výpočetní čas paralelních algoritmů se sekvenčními, používanými právě v sekci 6.5. Testování proběhlo na notebooku s procesorem Intel Core i7 a grafickou kartou NVIDIA GeForce GT 555M.

V tabulkách 7.1 a 7.2 je zaznamenána doba řešení této úlohy pomocí sekvenčních algoritmů a obou typů algoritmů paralelních. Čas výpočtů využívajících technologii CUDA byl měřen včetně alokace potřebných proměnných na grafické kartě. V případě metody MC bylo generováno vždy 10^5 náhodných pokusů.

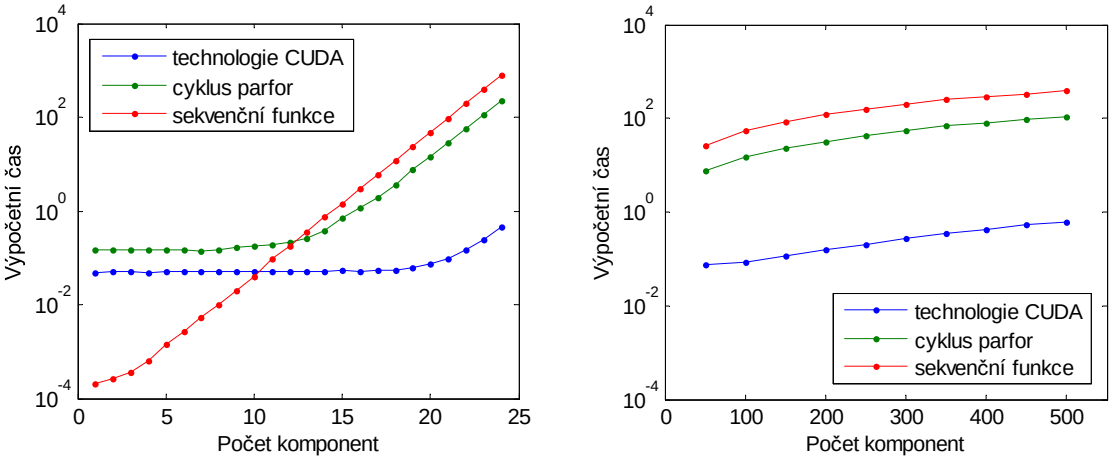
Počet komponent	5	10	15	20	21	22	23	24
Technologie CUDA	0,051	0,051	0,052	0,072	0,098	0,146	0,247	0,463
Cyklus `parfor`	0,149	0,177	0,697	14,64	29,23	55,51	112,1	223,1
Sekvenční funkce	0,001	0,041	1,451	47,82	95,72	193,9	392,5	780,3

Tabulka 7.1: Srovnání sekvenčních a paralelních algoritmů (analytické řešení)

Počet komponent	10	20	50	100	200	300	400	500
Technologie CUDA	0,057	0,057	0,073	0,084	0,155	0,269	0,426	0,628
Cyklus `parfor`	2,373	3,936	7,535	14,94	32,19	56,38	80,22	109,4
Sekvenční funkce	7,288	11,84	26,66	53,76	120,5	205,4	287,8	397,6

Tabulka 7.2: Srovnání sekvenčních a paralelních algoritmů (metoda MC)

Závislost výpočetního času (v sekundách) na počtu komponent znázorňují také grafy na obrázku 7.3. Pro lepší názornost bylo pro svislou osu zvoleno logaritmické měřítko.



(a) Analytické řešení

(b) Řešení metodou MC

Obrázek 7.3: Grafické znázornění rychlosti sekvenčních a paralelních algoritmů

Z tabulek a grafů je patrné, že použití cyklu `parfor` je výhodné až pro systémy o více než deseti komponentách. Řešením úlohy ve čtyřech vláknech lze dle očekávání dosáhnout přibližně čtyřikrát kratšího výpočetního času než v případě jednovláknové úlohy.

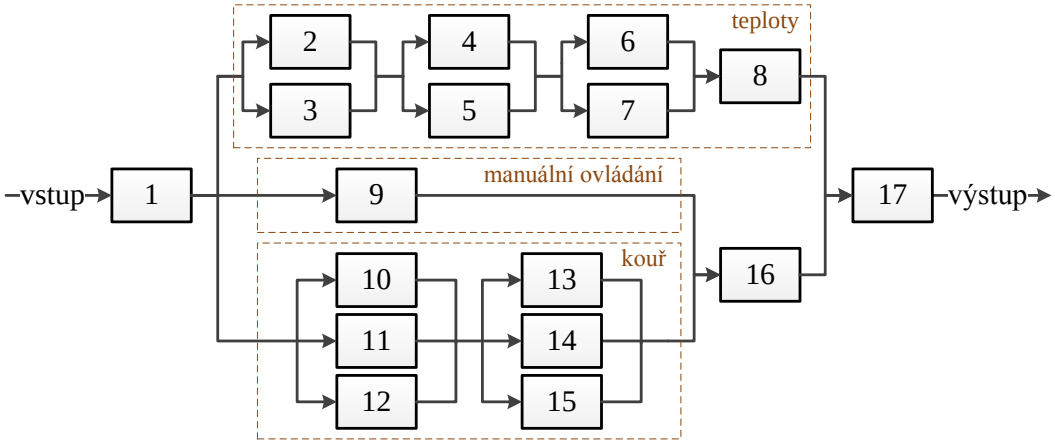
Požitím technologie CUDA bylo dosaženo výrazně lepších výsledků. Například pro 500 komponent byla tato úloha vyřešena přibližně 600krát rychleji než v případě sekvenčního algoritmu.

8 Řešení vybraných inženýrských úloh

V této kapitole je s využitím vytvořeného programu modelována pohotovost systémů a řešeny další související praktické úlohy. Cílem je především vytvořit si přibližný obrázek o tom, jaké úlohy z praxe je možné metodami uvedenými v kapitole 6 řešit.

Náhodné veličiny TTF a TTR jednotlivých komponent jsou vždy modelovány exponenciálním rozdělením.

8.1 Modelování pohotovosti systému požární ochrany



Obrázek 8.1: Systém požární ochrany

Obrázek 8.1 znázorňuje blokové schéma systému pro detekci požáru, jež byl vytvořen podle [8]. Systém lze primárně rozdělit na tři části, subsystém pro detekci kouře, subsystém pro detekci vysokých teplot a část umožňující manuální ovládání. Komponentami jsou všechny prvky systému, jejichž porucha může způsobit nefunkčnost celého systému. V následující tabulce je uveden význam jednotlivých komponent a jejich intenzita poruch a intenzita oprav, jednotkou je h^{-1}.

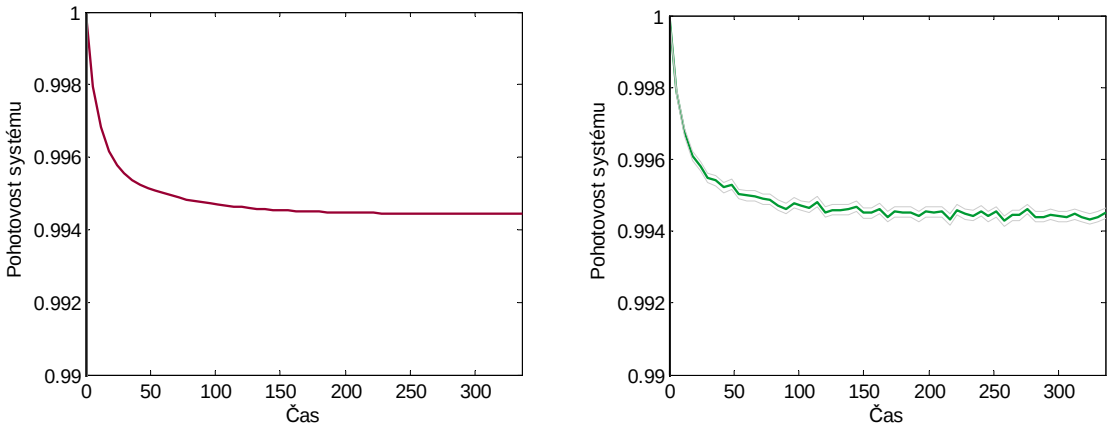
označení	popis komponenty	intenzita poruch	intenzita oprav
1	zdroj stejnosměrného proudu	$3 \cdot 10^{-5}$	0.018
$2,\ldots,7$	detektory kouře	$8.7 \cdot 10^{-4}$	0.025
8	požární hlásič	$1.1 \cdot 10^{-4}$	0.02
9	manuální ovládání	$2 \cdot 10^{-5}$	0.01
$10,\ldots,15$	teplotní detektory	$1.3 \cdot 10^{-3}$	0.03
16	tlakový spínač	$2.5 \cdot 10^{-4}$	0.11
17	spínací relé	$4.3 \cdot 10^{-4}$	0.11

Úkolem je modelovat pohotovost systému od uvedení do provozu v čase $t_0 = 0$ h do času $t_1 = 336$ h, tedy po dobu dvou týdnů.

8.1.1 Řešení

Má být modelována pohotovost systému v zadaném časovém intervalu. Určíme tedy pohotovost systému v časech od t_0 do t_1 s krokem například 6 h.

Podle zadaného schématu lze vytvořit odpovídající matici sousednosti. Tuto matici sousednosti a hodnoty hazardních funkcí stačí načíst do vytvořeného programu a úlohu poté vyřešit metodou Monte Carlo i analyticky. Výsledek analytického řešení a řešení metodou MC pro 10^6 náhodných pokusů znázorňují grafy na obrázku 8.2.



(a) Analyticky

(b) Metodou Monte Carlo

Obrázek 8.2: Modelování pohotovosti systému požární ochrany

Jinou možností je vytvořit systémovou funkci. K tomu lze využít možnost převodu matice sousednosti na systémovou funkci, kterou program poskytuje. Zde však bude vysvětlen postup ručního sestavení.

Nejdříve najdeme systémové funkce S_T a S_K subsystémů označených hesly „teploty“, resp. „kouř“.

$$\begin{aligned} S_T &= (b_2 + b_3) \cdot (b_4 + b_5) \cdot (b_6 + b_7) \cdot b_8 \\ S_K &= (b_{10} + b_{11} + b_{12}) \cdot (b_{10} + b_{11} + b_{12}) \end{aligned}$$

Systémová funkce celého systému má tvar

$$S(\mathbf{B}) = b_1 \cdot (S_T + (b_9 + S_K) \cdot b_{16}) \cdot b_{17}.$$

Tuto systémovou funkci taktéž můžeme zadat do programu a vyřešit úlohu oběma metodami. Graf řešení metodou MC není třeba uvádět, příliš se neliší od předchozího případu. Graf analytického řešení je pochopitelně stejný jako na obrázku 8.2a, liší se pouze výpočetní čas.

Jelikož se jedná o poměrně jednoduchý sério-paralelní systém, není obtížné najít ani normalizovaný tvar systémové funkce. Platí

$$\begin{aligned} S_T &= (1-(1-b_2)\cdot(1-b_3))\cdot(1-(1-b_4)\cdot(1-b_5))\cdot(1-(1-b_6)\cdot(1-b_7))\cdot b_8, \\ S_K &= (1-(1-b_{10})\cdot(1-b_{11})\cdot(1-b_{12}))\cdot(1-(1-b_{13})\cdot(1-b_{14})\cdot(1-b_{15})), \\ S(\mathbf{B}) &= b_1\cdot(1-(1-S_T)\cdot(1-((1-(1-b_9)\cdot(1-S_K))\cdot b_{16})))\cdot b_{17}. \end{aligned}$$

V tomto tvaru je funkce méně přehledná, lze však použít poznámku 4.2. Pravděpodobnost, že je i-tá komponenta v daném čase v provozu, vypočteme podle 6.1 a tyto pravděpodobnosti dosadíme do normalizované funkce. Tento způsob výpočtu pohotovosti je nejrychlejší, není však použitelný v případě složitějších systémů.

Pravdivostní tabulku nemá smysl vytvářet, systém se skládá ze 17 komponent, stavů je tedy celkem 131072.

Úloha byla řešena s pomocí vytvořeného programu i s využitím paralelních výpočtů. Doba řešení úlohy metodou MC (pro různé rozsahy výběrového souboru), analyticky i přímým dosazením je zaznamenána v tabulce 8.1, všechny časy jsou v sekundách.

Způsob zadání	Způsob řešení	Metoda MC				Analyticky
		10^3	10^4	10^5	10^6	
Systémová funkce	využit program	2.87	5.43	31.11	286.29	87.83
	cyklus `parfor`	12.75	13.87	21.92	95.28	42.19
Matice sousednosti	využit program	5.03	26.08	245,1	>1000	358.38
	cyklus `parfor`	12.92	20.09	88.61	771.22	123.43
	technologie CUDA	0.74	0.81	1.42	6.68	0.76
Dosazení do normalizované syst. funkce		-				0.024

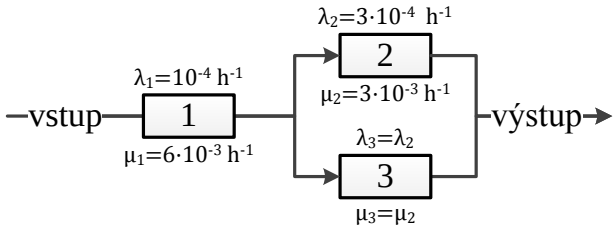
Tabulka 8.1: Doba řešení úlohy 8.1 v sekundách

Potvrdilo se, že použití cyklu `parfor` je výhodné až v případě složitějších výpočtů. Ukázalo se také, že v případě technologie CUDA je tuto úlohu vhodnější řešit analyticky než metodou MC, neboť přesného výsledku bylo dosaženo za pouhých 0.76 sekund.

8.2 Průměrný výkon elektrických generátorů

Tento příklad se zabývá výpočtem průměrného výkonu systému. Cílem je především poukázat na to, že již v případě systémů o nízkém počtu komponent je obtížné kvantifikovat průměrný výkon analyticky, a je výhodné použít metodu MC.

Schéma na obrázku 8.3 znázorňuje soustavu elektrických generátorů. Komponenta 1 reprezentuje přívod paliva, má intenzitu poruch λ_1 a intenzitu oprav μ_1. Komponenty 2 a 3 jsou elektrické generátory, každý o výkonu 100 MW, intenzitě poruch λ_2 a intenzitě oprav μ_2. Úkolem je spočíst průměrný výkon systému během šesti měsíců, tedy od času $t_0 = 0\,\text{h}$ do času $t_1 = 4320\,\text{h}$.



Obrázek 8.3: Soustava elektrických generátorů

8.2.1 Analytické řešení

Systémová funkce má tvar

$$S(\mathbf{B}) = b_1 \cdot (100b_2 + 100b_3),$$

vrací tak výkon soustavy v MW v závislosti na stavu systému. Je snadné si rozmyslet, že výkon bude nenulový pouze pro stavy $\mathbf{B} = (b_1, b_2, b_3) \in \{(1,0,1),(1,1,0),(1,1,1)\}$. Platí $S((1,0,1)) = S((1,1,0)) = 100\,\text{MW}$ a $S((1,1,1)) = 200\,\text{MW}$.

Pro určení pravděpodobného průměrného výkonu $Q_p(t_0;t_1)$ použijeme vzorec (6.6), který lze v tomto případě zjednodušit na tvar

$$Q_p(0;t_1) = \frac{100}{t_1} \cdot \left(\int_0^{t_1} P((1,0,1),t)\,\mathrm{d}t + \int_0^{t_1} P((1,1,0),t)\,\mathrm{d}t + 2 \cdot \int_0^{t_1} P((1,1,1),t)\,\mathrm{d}t \right).$$

Za použití vzorce (6.2) získáváme

$$P((1,0,1),t) = P((1,1,0),t) =$$

$$= \frac{\mu_1 + \lambda_1 \mathrm{e}^{-(\lambda_1+\mu_1)t}}{\lambda_1+\mu_1} \cdot \frac{\mu_2 + \lambda_2 \mathrm{e}^{-(\lambda_2+\mu_2)t}}{\lambda_2+\mu_2} \cdot \left(1 - \frac{\mu_2 + \lambda_2 \mathrm{e}^{-(\lambda_2+\mu_2)t}}{\lambda_2+\mu_2}\right),$$

$$P((1,1,1),t) = \frac{\mu_1 + \lambda_1 \mathrm{e}^{-(\lambda_1+\mu_1)t}}{\lambda_1+\mu_1} \cdot \left(\frac{\mu_2 + \lambda_2 \mathrm{e}^{-(\lambda_2+\mu_2)t}}{\lambda_2+\mu_2}\right)^2.$$

Po dosazení konkrétních hodnot a následné integraci vychází

$$\int_0^{4320} P((1,0,1),t)\,\mathrm{d}t = \int_0^{4320} P((1,1,0),t)\,\mathrm{d}t = 327.852, \int_0^{4320} P((1,1,1),t)\,\mathrm{d}t = 3564.74.$$

Platí tedy

$$Q_p(0;4320) = \frac{100}{4320} \cdot (2 \cdot 327.852 + 2 \cdot 3564.74) \doteq 180.2126$$

8.2.2 Řešení metodou Monte Carlo

Náhodnou veličinou X je zde pravděpodobný výkon systému v náhodně vygenerovaném čase $t \in (0;4320)$. Hledanou hodnotou je střední hodnota $\bar{X}$ této NV.

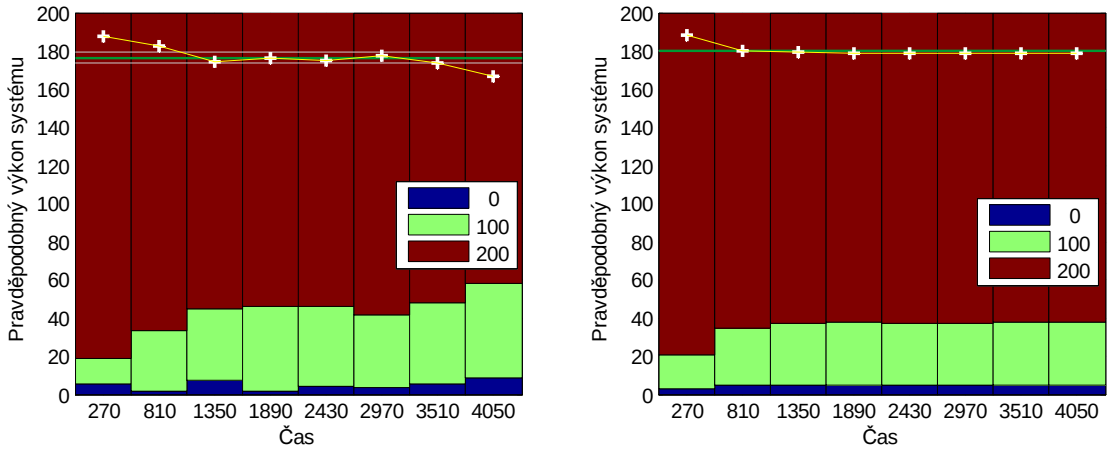
Algoritmus 6.5 pro výpočet průměrného výkonu je součástí programu. Stačí tedy správně zadat systémovou funkci a časový interval.

Výsledky pro různé rozsahy výběrového souboru jsou zaznamenány v tabulce 8.2.

Rozsah výběrového souboru	10^2	10^3	10^4	10^5	10^6
Odhad průměrného výkonu ($\bar{X}$)	182	180.8	180.78	179.935	180.1995
Hodnota ε při $\alpha = 0.05$	8.064	2.809	0.873	0.283	0.089
Skutečná chyba odhadu $\lvert Q_p - \bar{X} \rvert$	1.787	0.587	0.567	0.278	0.013
PRSD	2.261	0.793	0.246	0.080	0.025
Čas výpočtu v sekundách	0.002	0.01	0.10	0.98	9.72

Tabulka 8.2: Výsledky řešení úlohy 8.2 metodou MC

Obrázek 8.4 znázorňuje výsledky této úlohy graficky. Interval $(0;4320)$ je rozdělen na osm stejně velkých podintervalů, každému z nich v grafu odpovídá jeden sloupec. Jednotlivé sloupce jsou rozděleny na několik dílů podle relativní četnosti vypočtených hodnot pravděpodobného výkonu v časech, které padly do příslušného podintervalu. Žlutá lomená čára spojuje průměrné hodnoty výkonu v jednotlivých podintervalech, zelenou linkou je znázorněn průměrný výkon v celém intervalu $(0;4320)$. Šedé linky znázorňují interval, v němž se skutečná průměrná hodnota pravděpodobného výkonu nachází s pravděpodobností 0.95, v obrázku 8.4b již tyto linky nejsou rozeznatelné.

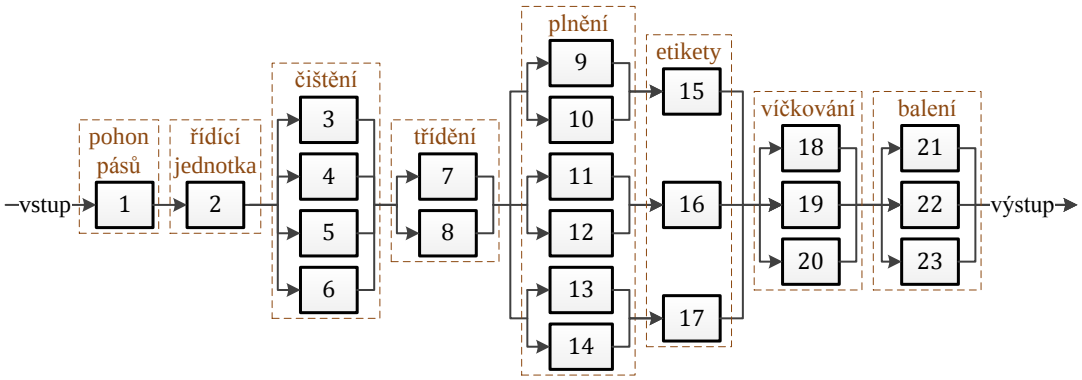


(a) Rozsah výběrového souboru 10^3

(b) Rozsah výběrového souboru 10^6

Obrázek 8.4: Grafické znázornění průměrného výkonu

8.3 Produkce systému pro plnění lahví



Obrázek 8.5: Schéma systému pro plnění lahví

Schéma na obrázku 8.5 znázorňuje zjednodušený systém sloužící k čištění lahví, plnění třemi druhy nápojů, etiketování, zavíčkování a balení. Následující tabulka uvádí MTTF a MTTR jednotlivých typů komponent v hodinách. U typů komponent, kde je to relevantní, je rovněž uvedeno, kolik lahví zvládne zpracovat (roztřídit, naplnit, ...) během jedné hodiny.

označení komponenty	MTTF	MTTR	lahve/h
1	3600	4	-
2	8640	16	-
$3,\ldots,6$	2160	20	300
$7,8$	5040	10	600
$9,\ldots,14$	2160	20	200
$15,\ldots,17$	2880	6	400
$18,\ldots,20$	2880	8	400
$21,\ldots,23$	5040	8	400

Dále je zadána systémová funkce $S(\mathbf{B})$, která pro každý stavový vektor vrací rychlost produkce tohoto systému v uvažovaném čase.

$$S(\mathbf{B}) = b_1 \cdot b_2 \cdot \min\left(\sum_{i=3}^{6} 300b_i, \sum_{i=7}^{8} 600b_i, \sum_{i=9}^{14} 200b_i, \sum_{i=15}^{17} 400b_i, \sum_{i=18}^{20} 400b_i, \sum_{i=21}^{23} 400b_i\right)$$

Úkol je podobný jako v předchozím příkladu, cílem je určit průměrnou produkci systému v průběhu jednoho měsíce, tedy od času $t_0 = 0\,\text{h}$ do času $t_1 = 720\,\text{h}$, s chybou menší než 0.1 (při hladině významnosti 0.05). Tento systém se skládá z vyššího počtu komponent, průměrná produkce je tak součtem 2^{23} určitých integrálů a vypočítat ji analyticky by bylo velice obtížné. Úloha tedy bude řešena pouze metodou Monte Carlo.

Dalším úkolem je určit průměrnou pohotovost systému během jednoho měsíce, je-li dáno, že systém je v provozu právě pro ty stavy, pro něž je výsledkem výše uvedené systémové funkce nenulová

hodnota. Tentokrát je požadováno, aby chyba odhadu byla nižší než 10^{-4} při téže hladině významnosti.

8.3.1 Řešení metodou Monte Carlo

Vstupem je nyní počet komponent, matice intenzit poruch a oprav a systémová funkce. Intenzitu poruch a oprav získáme jako převrácenou hodnotu MTTF a MTTR. Systémová funkce je zadávána formou textového řetězce, tedy v následující podobě:

```
b1*b2*min([300*(b3+b4+b5+b6),600*(b7+b8),200*(b9+b10+b11+b12+b13+
b14),400*(b15+b16+b17),400*(b18+b19+b20),400*(b21+b22+b23)])
```

Průměrný výkon má být odhadnut s chybou nižší než 0.1. Počet náhodných pokusů, které je nutné provést pro dosažení této přesnosti, vypočteme pomocí vzorce 3.4. Směrodatnou odchylku σ_X nahradíme odhadem výběrové směrodatné odchylky s náhodné veličiny X, která značí pravděpodobný výkon systému v náhodně vygenerovaném čase $t \in (0;720)$. Pro získání tohoto odhadu provedeme předvýpočet o nízkém počtu náhodných pokusů, tedy například 10^4.

Odhad výběrové směrodatné odchylky s byl stanoven na 114.2, tento předvýpočet trval 0.22 sekund. Do vzorce dále dosadíme $\varepsilon = 0.1$ a $\alpha = 0.05$, získáme tak odhad počtu náhodných pokusů, které je třeba provést,

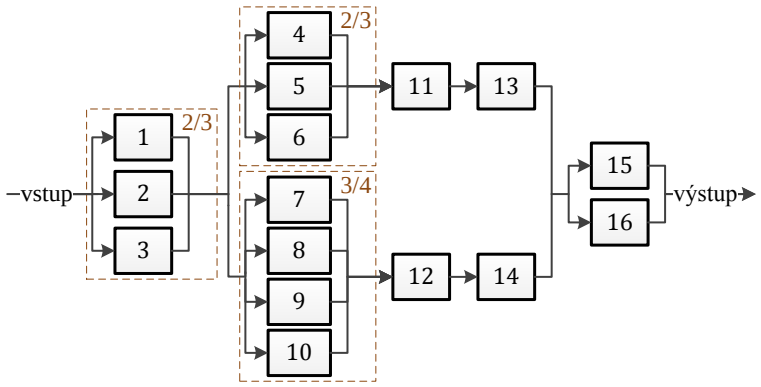
$$N = \left(\frac{\sigma_X}{\varepsilon} \cdot z_{1-\frac{\alpha}{2}}\right)^2 \approx \left(\frac{114.2}{0.1} \cdot z_{0.975}\right)^2 \doteq 5009000.$$

Simulace metodou MC byla tedy realizována pro N náhodných pokusů. Výpočet průměrné hodnoty pravděpodobného výkonu trval přibližně 109 sekund. Výsledkem je hodnota 1166.24, chyba odhadu s pravděpodobností 0.95 není vyšší než 0.0989. Touto průměrnou rychlostí je systém schopen během jednoho měsíce naplnit 839692 lahví.

V případě určování průměrné pohotovosti byl také nejprve proveden předvýpočet, na jeho základě byla výběrová směrodatná odchylka s odhadnuta číslem 0.054 a počet náhodných pokusů, které je třeba realizovat, stanoven na 1120000. Výpočet průměrné pohotovosti při tomto počtu náhodných pokusů trval $35,4$ sekund. Výsledkem je hodnota $99,704\%$, systém tedy bude v provozu pravděpodobně 717 hodin a 52 minut.

Pro srovnání si všimněme, že pokud by systém pracoval zcela bez poruch, bylo by za 720 hodin nepřetržitého provozu zpracováno celkem 864000 lahví.

8.4 Proces výroby kovů



Obrázek 8.6: Průmyslový systém výroby kovů, podle [7, str. 133]

Tento příklad slouží k vysvětlení jedné z možností vytvořeného programu. Je-li systém zadán pomocí matice sousednosti, lze navíc vytvořit tzv. „bloky *k*/*m*“. Jedná se o skupiny *m* komponent, z nichž musí být alespoň *k* v provozu, aby byla v provozu celá skupina. Systém na obrázku 8.6 obsahuje tři takové bloky. Blok komponent $1,\ldots,3$ znázorňuje tři zdroje napájení, z nichž musí být v provozu alespoň dva. Další dva bloky reprezentují skupiny pecí, z nichž musí být v provozu vždy vyznačený počet.

Systém stačí zadat do programu pomocí běžné matice sousednosti, bez ohledu na vyznačené bloky. Tuto matici znázorňuje graf na obrázku 8.7a. Dále je třeba zvolit možnost „Správa bloků“, přidat tyto tři skupiny komponent a zadat, kolik komponent ve které skupině musí být minimálně v provozu.

V následující tabulce jsou uvedeny hodnoty MTTF a MTTR jednotlivých komponent v hodinách. Hodnoty jsou převzaty z [7, str. 145].

označení komponenty	MTTF	MTTR
$1,2,3$	2300	4
$3,\ldots,10$	860	24
$11,12$	1250	24
$13,14$	380	8
$15,16$	900	12

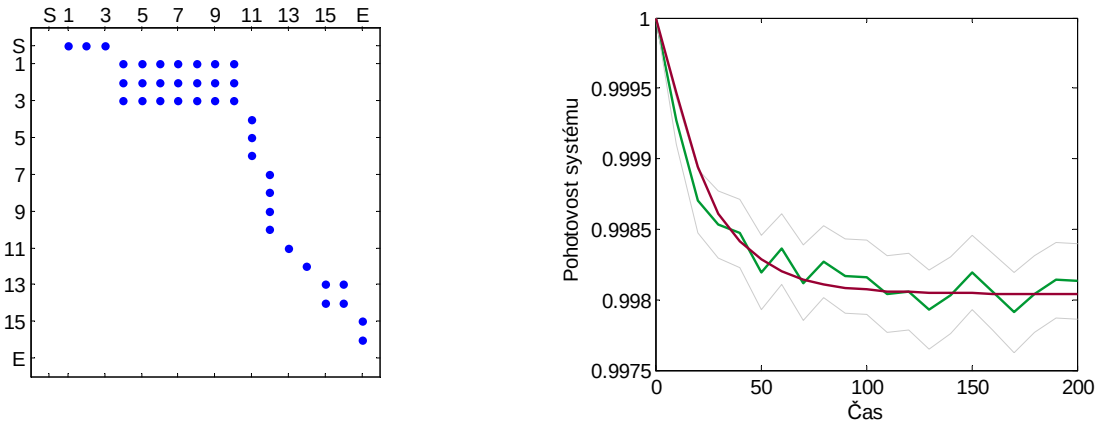
Úkolem je modelovat pohotovost tohoto systému od spuštění v čase $t_0 = 0\,\text{h}$ do času $t_1 = 200\,\text{h}$ a metodou MC určit, po jakou část tohoto časového intervalu bude systém pravděpodobně v provozu.

8.4.1 Řešení

Pohotovost systému byla modelována v zadaném intervalu s krokem 10 h. Pro odhad pohotovosti v každém z těchto časů metodou MC bylo použito 10^5 náhodných pokusů. Pro určení průměrné hodnoty pohotovosti bylo generováno celkem 10^6 náhodných pokusů. Graf na obrázku 8.7b znázorňuje

křivku pohotovosti určenou analyticky (červená křivka) i její odhad metodou MC (zelená lomená čára).

Průměrná pohotovost systému na zadaném intervalu (o délce 200 h) byla odhadnuta číslem 0.99827, odchylka od přesné hodnoty je s pravděpodobností 0.95 menší než $8 \cdot 10^{-5}$. Systém tedy bude v provozu pravděpodobně $0.99827 \cdot 200 \doteq 199.654$ h, tedy přibližně 199 hodin a 39 minut.



(a) Znázornění matice sousednosti

(b) Pohotovost systému výroby kovů

Obrázek 8.7: Řešení příkladu v sekci 8.4

9 Závěr

Hlavním cílem této bakalářské práce bylo seznámení se simulační metodou Monte Carlo a její aplikace na problematiku určování pohotovosti systému. Nejprve však bylo nutné věnovat se teorii, tedy uvést základní poznatky z oblasti pravděpodobnosti a statistiky, definovat systémy s nezávislými prvky a objasnit princip metody Monte Carlo a základy teorie spolehlivosti.

V praktické části byly představeny algoritmy pro simulační i analytický výpočet pohotovosti systémů s nezávislými prvky v období stabilního života. Tyto algoritmy byly navíc modifikovány tak, aby systém mohl být místo systémové funkce zadáván také pravdivostní tabulkou, nebo pomocí matice sousednosti, která popisuje jeho strukturu. Vedle modelování pohotovosti systému byl určován také tzv. výkon systému. Jedná se o úlohu příbuznou, v podstatě jde o určení pravděpodobné hodnoty systémové funkce, která vypovídá o jisté vlastnosti systému, jíž lze vyjádřit číselně.

Algoritmy byly implementovány v jazyce Matlab a porovnávány z hlediska časové složitosti. Bylo ověřeno, že v případě analytického řešení roste výpočetní čas exponenciálně vzhledem k počtu komponent, což znemožňuje analytické řešení pohotovosti rozsáhlejších systémů. Výpočetní čas metody Monte Carlo roste vzhledem k počtu komponent přibližně lineárně. Výhodou simulačního řešení je rovněž to, že lze předem určit, kolik náhodných pokusů je třeba provést, aby bylo dosaženo požadované přesnosti při jisté hladině významnosti.

Pro snadnější řešení pohotovosti systému a souvisejících úloh byl vytvořen program s grafickým uživatelským rozhraním, který používá zmíněné simulační i analytické algoritmy.

Algoritmy pro určení pohotovosti systému zadaného maticí sousednosti byly dále optimalizovány pomocí paralelních výpočtů. Využitím paralelního cyklu v prostředí Matlab byl výpočetní čas dle očekávání zkrácen přibližně tolikrát, kolik jader měl procesor. Výraznějšího zrychlení bylo dosaženo použitím technologie CUDA, která umožňuje provádění paralelních výpočtů na grafické kartě. Je třeba zmínit, že byl pro tento účel algoritmus značně upraven, například použitím bitových operací.

Poslední část byla věnována několika konkrétním příkladům z inženýrské praxe a návodu, jak k jejich řešení využít zmíněný program.

Téma této práce nabízí prostor k dalšímu rozšiřování. Například je možné zkoumat pohotovost systémů v období časných poruch a v období stárnutí, zabývat se komponentami, které nabývají většího počtu stavů, nebo průběžně udržovanými systémy. Součástí predikce chování systému je také odhalování jeho slabých míst, tedy identifikace komponent, které nejčastěji zapříčiňují výpadky, a následné navyšování spolehlivosti systému.

10 Reference

[1] DUBI, A. *Monte Carlo Applications in Systems Engineering.* Wiley, 1999. ISBN 0-471-98172-9.

[2] BRIŠ, Radim. *Inovační metody pro ocenění spolehlivosti prvků a systémů.* 1. vyd. Ostrava: VŠB - TECHNICKÁ UNIVERZITA OSTRAVA, 2007. ISBN 978-80-248-1596-1.

[3] LITSCHMANNOVÁ, Martina. *Vybrané kapitoly z pravděpodobnosti* [online]. VŠB – TU Ostrava, Fakulta elektrotechniky a informatiky, 2011. Dostupné na <mi21.vsb.cz> [citováno 14. 3. 2013].

[4] LITSCHMANNOVÁ, Martina. *Úvod do statistiky* [online]. VŠB – TU Ostrava, Fakulta elektrotechniky a informatiky, 2011. Dostupné na <mi21.vsb.cz> [citováno 18. 3. 2013].

[5] FABIAN, František, KLUIBER, Zdeněk. *METODA MONTE CARLO a možnosti jejího uplatnění.* 1. vyd. Praha: PROSPEKTRUM, 1998. ISBN 80-7175-058-1.

[6] RUBINSTEIN, Reuven Y., KROESE, Dirk P. *Simulation and the Monte Carlo method*. Wiley, 2008. ISBN 978-0-470-17794-5.

[7] DUBI, A. *Predictive modeling and simulation for maximizing system performance.* JMO INK Publishing, 2006. ISBN 0-9739807-1-0.

[8] RAUSAND, Marvin, HØYLAND, Arnljot. *System reliability theory.* 2nd edition. Wiley, 2004. ISBN 0-471-47133-X

[9] *Los Alamos Science* [online]. No. 15, 1987. Los Alamos National Laboratory. Dostupné na <http://la-science.lanl.gov/lascience15.shtml> [citováno 24. 3. 2013].

[10] *CUDA C Programming Guide* [online]. PG-02829-001_v5.0, October 2012. Dostupné na <http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf> [citováno 17. 4. 2013].

[11] DIVIŠ, Zdeněk, ZDRÁLEK, Jaroslav, CHMELÍKOVÁ, Zdeňka. *Logické obvody.* 2. vyd. Ostrava: VŠB - TECHNICKÁ UNIVERZITA OSTRAVA, 2008. ISBN 978-80-248-1724-8

[12] KOZUBEK, T., LAMPART, M. *Integrální transformace* [online]. VŠB – TU Ostrava, Fakulta elektrotechniky a informatiky, 2012. Dostupné na <mi21.vsb.cz> [citováno 29. 3. 2013].

A Zdrojové kódy

A.1 Výpočet výkonu systému

A.1.1 Analytický výpočet pravděpodobného výkonu systému

```
function [ Q ] = Vykon_analyticky( S, n, p )
%   S   systémová funkce (handle) udávající výkon systému
%   n   počet komponent systému
%   p   vektor pravděpodobností provozu jednotlivých komponent
Q=0;                        % pravděpodobný výkon systému
for j=1:2^n
    B=r2B(j,n);             % f-ce pro určení j-tého st. vektoru
    Q=Q+Pr(B,n,p)*S(B);     % inkrementace výstupní proměnné
end
```

Výpis A.1: Zdrojový kód algoritmu 6.4

A.1.2 Výpočet pravděpodobného výkonu systému metodou Monte Carlo

```
function [ Q, epsilon, PRSD ] = Vykon_MC( S, n, p, N, alpha )
%   S   systémová funkce (handle) udávající výkon systému
%   n   počet komponent systému
%   p   vektor pravděpodobností provozu jednotlivých komponent
%   N   rozsah výběrového souboru
q=0;    % součet prvních mocnin NV
m=0;    % součet druhých mocnin NV
for i=1:N
    B=rand(1,n)<p;          % vygenerován náhodný stavový vektor
    vykon=S(B);             % výkon systému ve stavu B
    q=q+vykon;
    m=m+vykon*vykon;
end
Q=q/N;                      % pravděpodobný výkon systému
s=sqrt((m-q*q/N)/(N-1));    % výběrová směrodatná obchylka
epsilon=s*norminv(1-alpha/2)/sqrt(N);
PRSD=100*s/(Q*sqrt(N));
```

Výpis A.2: Zdrojový kód algoritmu 6.5

A.1.3 Výpočet průměrného výkonu systému metodou Monte Carlo

```
function [ Qp, epsilon, PRSD ] = VykonPrum_MC( S, n, intenzita, N, od, do )
% Vypočte průměrný výkon v intervalu <od,do>.
%   S   systémová funkce (handle) udávající výkon systému
%   n   počet komponent systému
```

```
%   intenzita   vektory intenzity poruch a oprav
%   N   rozsah výběrového souboru
q=0;    % součet prvních mocnin NV
m=0;    % součet druhých mocnin NV
lambda=intenzita(1,:);  % vektor intenzity poruch jednotlivých komponent
mu=intenzita(2,:);      % vektor intenzity oprav jendotlivých komponent
for i=1:N
    time=rand()*(do-od)+od; % vybrán náhodný čas z intervalu <od,do>
    p=Pit(lambda,mu,time);  % určeny pravděpodobnosti provozu komponent
    B=rand(1,n)<p;          % vygenerován náhodný stavový vektor
    vykon=S(B);             % výkon systému ve stavu B
    q=q+vykon;
    m=m+vykon*vykon;S
end
Qp=q/N;     % pravděpodobný průměrný výkon systému v intervalu <od,do>
s=sqrt((m-q*q/N)/(N-1));    % výběrová směrodatná obchylka
epsilon=s*norminv(1-alpha/2)/sqrt(N);
PRSD=100*s/(Qp*sqrt(N));
```

Výpis A.3: Zdrojový kód algoritmu 6.6

A.2 Rozhodnutí o průchodnosti sytému

```
function [ pruchozi ] = PruchodB( A, B )
% Rozhodne o stavu systému daného matici sousednosti A a stavovým vektorem B.
vyber=logical([ 1 B 1 ]);
% ořeže matici A o řádky a sloupce příslušné nefunkčním komponentám
A_orez=A(vyber,vyber);
% volá funkci pro zjištění, zda existuje cesta ze vstupu na výstup
pruchozi=Pruchodnost(A_orez);
```

Výpis A.4: Zdrojový kód funkce `PruchodB(A,B)`

```
function [ pruchozi ] = Pruchodnost( A )
%   Rozhoduje, zda v systému daném maticí sousednosti A existuje
%   cesta ze vstupu na výstup.
n=length(A);    % počet pruků systému (komponenty + vstup + výstup)
stare=false(1,n);
stare(1)=true;  % všechny již navštívené komponenty
nove=A(1,:);    % pouze naposledy navštívené komponenty
pruchozi=false; % příznak průchodnosti systému
while any(nove)>0       % nebyly navštíveny nové komponenty -> konec (0)
    suma=any(A(nove,:),1);
    if suma(end)==true  % navštíven koncový prvek -> konec (1)
        pruchozi=true;
        return
    end
```

```
15      nove=suma&(~stare);
16      stare=stare|suma;
17  end
```

Výpis A.5: Zdrojový kód funkce `Pruchodnost(A)`

A.3 Paralelizace

A.3.1 Použití paralelního cyklu v Matlabu

```
1  function [ res, epsilon, PRSD, time ] = Parfor_MC( A, n, p, N, alpha )
2  %   A   matice sousednosti
3  %   n   počet komponent systému
4  %   p   vektor pravděpodobností provozu jednotlivých komponent
5  %   N   rozsah výběrového souboru
6  tic
7  pole=zeros(1,N);
8  parfor i=1:N
9      B=rand(1,n)<p;                  % vygenerován náhodný stavový vektor
10     if PruchodB(A,B)==false         % vyhodnocena průchodnost
11         pole(i)=1;
12     end
13 end
14 n_0=sum(pole); % počet stavových vektorů, pro které byl systém mimo provoz
15 time=toc;
16 res=1-n_0/N;                        % hledaná pohotovost
17 s=sqrt((n_0-n_0*n_0/N)/(N-1));      % výběrová směrodatná obchylka
18 epsilon=s*norminv(1-alpha/2)/sqrt(N);
19 PRSD=100*s/((n_0/N)*sqrt(N));
```

Výpis A.6: Simulační algoritmus, využit cyklus `parfor`

```
1  function [ res ] = Parfor_A( A, n, p )
2  %   A   matice sousednosti
3  %   n   počet komponent systému
4  %   p   vektor pravděpodobností provozu jednotlivých komponent
5  U=zeros(1,4);
6  parfor i=1:4                        % úloha běží ve 4 vláknech
7      for j=((i-1)*2^(n-2)+1):(i*2^(n-2)) % v každém vlákně 1/4 stavů
8          B=r2B(j,n);                 % f-ce pro určení j-tého st. vektoru
9          if PruchodB(A,B)==false     % vyhodnocena průchodnost
10             U(i)=U(i)+Pr(B,n,p);    % inkrementace nedostupnosti
11         end
12     end
13 end
14 res=1-sum(U);                       % pohotovost systému
```

Výpis A.7: Analytický algoritmus, využit cyklus `parfor`

A.3.2 Práce s CUDA v prostředí Matlab

```
function [ res, epsilon, PRSD, time ] = CUDA_MC( M, p, N, alpha )
% M     matice sousednosti
% p     vektor pravděpodobností provozu jednotlivých komponent
% N     rozsah náhodného výběru, upraví se dle velikosti bloků a gridu
n=length(M)-2;          % počet komponent systému
A=prevod_mat(M,32);     % převod matice sousednosti na vektor uint32
reset(gpuDevice());     % uvolnění paměti GPU
k=parallel.gpu.CUDAKernel('Pohot_MC.ptx','Pohot_MC.cu'); % vytvořen kernel
tic;
b_size=min(N,1000);                     % počet vláken na blok
g_size=min(ceil(N/b_size),5000);        % rozměry mřížky bloků
N=b_size*g_size;                        % úprava počtu náhodných pokusů
k.ThreadBlockSize=[b_size 1 1];         % nastavení rozměrů bloku
k.GridSize=[g_size 1];                  % nastavení rozměrů mřížky
pole=gpuArray.rand([1,N*n],'single');   % vytvoření pole náh. čísel na GPU
v=gpuArray(zeros(1, N, 'int32'));       % alokován výstupní vektor
P=single(gpuArray(p));                  % vektor p -> GPU
[~,vystup,~,~]=feval(k,n,pole,v,P,A);   % spuštění kernelu
n_0=gather(sum(vystup));    % počet stavů, pro které je systém mimo provoz
res=1-n_0/N;                            % výsledná pohotovost
time=toc;
s=sqrt((n_0-n_0*n_0/N)/(N-1));          % výběrová sm. obchylka
epsilon=s*norminv(1-alpha/2)/sqrt(N);
PRSD=100*s/((n_0/N)*sqrt(N));
```

Výpis A.8: Práce s CUDA v Matlabu (simulační algoritmus)

```
function [ res, time ] = CUDA_A( M, p )
% M     matice sousednosti
% p     vektor pravděpodobností provozu jednotlivých komponent
n=length(M)-2;              % počet komponent systému
A=prevod_mat(M,32);         % převod matice sousednosti na vektor uint32
reset(gpuDevice());         % uvolnění paměti GPU
k=parallel.gpu.CUDAKernel('Pohot_A.ptx','Pohot_A.cu');  % vytvořen kernel
tic;
b_size=min(1024,2^n);                   % počet vláken na blok
g_size=min((2^n)/b_size,2^14);          % rozměry mřížky bloků
davka=2^n/(b_size*g_size);
k.ThreadBlockSize=[b_size 1 1];         % nastavení rozměrů bloku
k.GridSize=[g_size 1];                  % nastavení rozměrů mřížky
v=gpuArray(zeros(1, 2^n, 'single'));    % alokován výstupní vektor na GPU
P=single(gpuArray(p));                  % vektor p -> GPU
[vystup,~]=feval(k,n,v,P,A,davka);
res=1-gather(sum(vystup));              % výsledná pohotovost
time=toc;
```

Výpis A.9: Práce s CUDA v Matlabu (analytický algoritmus)

A.3.3 Zdrojové kódy CUDA

```
__global__ void mc(int K, float *pole, int *v, float *P, unsigned int *A)
{
    int idx = blockIdx.x * blockDim.x + threadIdx.x;    // číslo vlákna
    const int CH=32;
    int rozsah = (K+2-1)/CH+1;  // kolik int32 je potreba
    unsigned int B[16];         // naposledy navštívené komponenty
    unsigned int stare[16];     // již navštívené nebo nefunkční komp.
    unsigned int suma[16];
    int pruchozi=0;             // příznak průchodnosti systému
    int pokracovat=1;           // příznak vynulován, je-li B nulový
    for(int i=1; i<K+1; i++)    // vygenerovani náhodného
    {                           // stavového vektoru
        if(pole[idx*K+i-1]<P[i-1])
            B[i/CH]|=1<<(i%CH);
    }
    B[rozsah-1]|=1<<((K+1)%CH); // zapsání čísla 1 na pozici výstupu
    for(int i=0; i<rozsah; i++)
    {   stare[i]=~B[i];         // nefunkční komponenty
        B[i]=B[i]&A[i];         // funkční komponenty dostupné ze vstupu
        stare[i]|=B[i]; }       // přidány již navštívené komponenty
    while(pokracovat==1 && pruchozi==0)
    {   for(int i=0; i<rozsah; i++)
            suma[i]=0;
        for(int i=1; i<K+1; i++) // do proměnné suma zaznamenány nově
        {                        // přístupné komponenty
            if((B[i/CH]&(1<<(i%CH)))>0)
            {   for(int j=0; j<rozsah; j++)
                    suma[j]=suma[j]|A[rozsah*i+j]; }
        }
        if ( ( suma[rozsah-1]&(1<<((K+1)%CH)) )>0  )
            pruchozi=1;                     // nalezen výstupní prvek
        for(int i=0; i<rozsah; i++)
        {   B[i]=(suma[i]^stare[i])&suma[i];// odebrány již navštívené
            stare[i]=stare[i]|B[i]; }       // aktualizace již navštívených
        pokracovat=0;       // pokud byly navštíveny nové komponenty -> 1
        for(int i=0; i<rozsah; i++)
        {   if(B[i]>0)
                pokracovat=1; }
    }
    v[idx]=1-pruchozi;
}
```

Výpis A.10: CUDA kernel (simulační algoritmus)

```
__global__ void analyt(int K, float *v, float* P, unsigned int *A, int D)
{
```

```
3       int id = blockIdx.x * blockDim.x + threadIdx.x; // číslo vlákna
4       unsigned int B;
5       unsigned int nove;
6       unsigned int stare;
7       unsigned int suma;
8       int pruchozi;
9       v[id]=0;
10      for(int d=0;d<D;d++) // v 1 vlákně může být vyhodnoceno více stavů
11      {
12          B=(id*D+d)<<1;              // stavový vektor
13          nove=B;                     // naposledy navštívené komponenty
14          nove|=1<<K+1;               // zapsání čísla 1 na pozici výstupu
15          stare=~B;                   // nefunkční komponenty
16          nove=nove&A[0];             // funkční komponenty dostupné ze vstupu
17          stare|=nove;                // přidány již navštívené komponenty
18          pruchozi=0;                 // příznak průchodnosti systému
19          while (nove!=0 && pruchozi==0)
20          {
21              suma=0;
22              for(int i=1;i<K+1;i++) // do proměnné suma zaznamenány nově
23              {                      // přístupné komponenty
24                  if ((nove&(1<<i))>0)
25                  {
26                      suma=suma|A[i];
27                  }
28              }
29              if ((suma&(1<<K+1))>0)
30                  pruchozi=1; // mezi novými komponentami nalezena výstupní
31              nove=(suma^stare)&suma; // odebrány již navštívené
32              stare=stare|nove;       // aktualizace již navštívených
33          }
34          if(pruchozi==0)             // pokud systém není průchozí,
35          {                           // určena pravděpodost nastání stavu
36              float prod=1;
37              for(int i=1; i<(K+1); i++)
38              {
39                  if ((B&(1<<i))>0)
40                      prod=prod*P[i-1];
41                  else
42                      prod=prod*(1-P[i-1]);
43              }
44              v[id]+=prod;
45          }
46      }
47  }
```

Výpis A.11: CUDA kernel (analytický algoritmus)

B Testování algoritmů pro výpočet pohotovosti systému

n	2^n	$A(t)$	$U(t)$	τ_F [s]	τ_M [s]
1	2	0,999	0,001	0,000718409	0,000206288
2	4	0,998001	0,001999	0,000530083	0,000270945
3	8	0,997002999	0,002997001	0,000311482	0,000350997
4	16	0,996005996	0,003994004	0,000395638	0,000649653
5	32	0,99500999	0,00499001	0,000563438	0,001442991
6	64	0,99401498	0,00598502	0,000932906	0,002626839
7	128	0,993020965	0,006979035	0,001637973	0,005277796
8	256	0,992027944	0,007972056	0,003181527	0,010211265
9	512	0,991035916	0,008964084	0,006034638	0,020140296
10	1024	0,99004488	0,00995512	0,011270815	0,040868155
11	2048	0,989054835	0,010945165	0,024118538	0,092683002
12	4096	0,98806578	0,01193422	0,045924824	0,181991533
13	8192	0,987077715	0,012922285	0,091441693	0,364248877
14	16384	0,986090637	0,013909363	0,177848877	0,732124182
15	32768	0,985104546	0,014895454	0,359183579	1,451981013
16	65536	0,984119442	0,015880558	0,739096857	2,938267094
17	131072	0,983135322	0,016864678	1,482807441	5,905300321
18	262144	0,982152187	0,017847813	2,931839897	11,82186581
19	524288	0,981170035	0,018829965	6,020835407	23,77697498
20	1048576	0,980188865	0,019811135	12,87711559	47,8219628
21	2097152	0,979208676	0,020791324	24,43293188	95,72386145
22	4194304	0,978229467	0,021770533	48,93312174	193,9155361
23	8388608	0,977251238	0,022748762	99,80701629	392,5519846
24	16777216	0,976273987	0,023726013	201,1574471	780,3699182

Tabulka B.1: Test analytického algoritmu

n	$\bar{X}$	$1-\bar{X}$	$A(t)$ podle (6.7)	$\lvert 1-\bar{X}-A(t)\rvert$	ε ($\alpha = 0.05$)	*PRSD*	τ_F [s]
5	0,00455	0,99545	0,99501	0,00044	0,00042	4,68%	0,6499
10	0,00987	0,99013	0,99004	0,00009	0,00061	3,17%	0,6537
15	0,01458	0,98542	0,9851	0,00032	0,00074	2,60%	0,6956
20	0,01998	0,98002	0,98019	0,00017	0,00087	2,21%	0,7181
25	0,02564	0,97436	0,9753	0,00094	0,00098	1,95%	0,731
30	0,02876	0,97124	0,97043	0,00081	0,00104	1,84%	0,7675
35	0,03466	0,96534	0,96559	0,00025	0,00113	1,67%	0,7565
40	0,03914	0,96086	0,96077	0,00009	0,0012	1,57%	0,7766
45	0,04339	0,95661	0,95598	0,00063	0,00126	1,48%	0,8192
50	0,04955	0,95045	0,95121	0,00076	0,00135	1,38%	0,8872
60	0,05778	0,94222	0,94174	0,00048	0,00145	1,28%	0,9182
70	0,06728	0,93272	0,93236	0,00036	0,00155	1,18%	0,8753
80	0,07648	0,92352	0,92308	0,00044	0,00165	1,10%	0,9969
90	0,08659	0,91341	0,91389	0,00048	0,00174	1,03%	0,9383
100	0,09523	0,90477	0,90479	0,00002	0,00182	0,97%	0,9702
150	0,1431	0,8569	0,86064	0,00374	0,00217	0,77%	1,1253
200	0,18109	0,81891	0,81865	0,00026	0,00239	0,67%	1,2583
250	0,22276	0,77724	0,7787	0,00146	0,00258	0,59%	1,4173
300	0,2601	0,7399	0,74071	0,00081	0,00272	0,53%	1,6222
350	0,29559	0,70441	0,70456	0,00015	0,00283	0,49%	1,8062
400	0,33026	0,66974	0,67019	0,00045	0,00291	0,45%	1,996
450	0,3664	0,6336	0,63748	0,00388	0,00299	0,42%	2,1984
500	0,39436	0,60564	0,60638	0,00074	0,00303	0,39%	2,3545
600	0,45128	0,54872	0,54865	0,00007	0,00308	0,35%	2,6873
700	0,50443	0,49557	0,49641	0,00084	0,0031	0,31%	3,0608
800	0,54949	0,45051	0,44915	0,00136	0,00308	0,29%	3,471

Tabulka B.2: Test simulačního algoritmu, systém zadán systémovou funkcí

n	$\bar{X}$	$1-\bar{X}$	$A(t)$ podle (6.7)	$\lvert 1-\bar{X}-A(t)\rvert$	ε ($\alpha = 0.05$)	*PRSD*	τ_M [s]
5	0,00519	0,99481	0,99501	0,0002	0,00045	4,38%	5,008
10	0,00968	0,99032	0,99004	0,00028	0,00061	3,20%	7,288
15	0,01517	0,98483	0,9851	0,00027	0,00076	2,55%	9,571
20	0,02001	0,97999	0,98019	0,0002	0,00087	2,21%	11,84
25	0,02426	0,97574	0,9753	0,00044	0,00095	2,01%	14,10
30	0,02988	0,97012	0,97043	0,00031	0,00106	1,80%	16,55
35	0,03465	0,96535	0,96559	0,00024	0,00113	1,67%	19,33
40	0,03928	0,96072	0,96077	0,00005	0,0012	1,56%	21,98
45	0,04493	0,95507	0,95598	0,00091	0,00128	1,46%	24,35
50	0,04875	0,95125	0,95121	0,00004	0,00133	1,40%	26,66
60	0,05688	0,94312	0,94174	0,00138	0,00144	1,29%	31,54
70	0,06587	0,93413	0,93236	0,00177	0,00154	1,19%	36,96
80	0,07705	0,92295	0,92308	0,00013	0,00165	1,09%	42,86
90	0,08761	0,91239	0,91389	0,0015	0,00175	1,02%	47,85
100	0,09614	0,90386	0,90479	0,00093	0,00183	0,97%	53,76
150	0,14015	0,85985	0,86064	0,00079	0,00215	0,78%	86,43
200	0,18224	0,81776	0,81865	0,00089	0,00239	0,67%	120,5
250	0,22221	0,77779	0,7787	0,00091	0,00258	0,59%	156,3
300	0,2606	0,7394	0,74071	0,00131	0,00272	0,53%	205,4
350	0,29512	0,70488	0,70456	0,00032	0,00283	0,49%	249,1
400	0,32842	0,67158	0,67019	0,00139	0,00291	0,45%	287,8
450	0,36314	0,63686	0,63748	0,00062	0,00298	0,42%	336,9
500	0,39227	0,60773	0,60638	0,00135	0,00303	0,39%	397,6

Tabulka B.3: Test simulačního algoritmu, systém zadán maticí sousednosti

C Příloha na CD

Přiložené CD obsahuje

- zdrojové kódy programu popsaného v sekci 7.1,
- uživatelskou příručku k tomuto programu,
- zdrojové kódy ostatních algoritmů zmiňovaných v této práci,
- elektronickou verzi této práce.